UIUCDCS-R-75-706

THE DESIGN AND IMPLEMENTATION OF A CALCULATOR TERMINAL

by

John Patrick Sheehan

March, 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-75-706

THE DESIGN AND IMPLEMENTATION OF A CALCULATOR TERMINAL

by

John Patrick Sheehan

March, 1975

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Submitted in partial fulfillment for the requirements for the degree of Master of Science in Electrical Engineering, March 1975.

ACKNOWLEDGEMENTS

The author would like to thank Professor Michael Faiman for his advice and support throughout the development of this project and the writing of this thesis. Special thanks also goes to Professor Donald Gillies who provided the computer so that the project could be tested and verified. Greg Chesson, Paul Krabee, and Ian Stocks have invested an amount of time and effort in helping me with the various computer software and hardware difficulties. I am deeply indebted to them.

I am very grateful to Ms. Evelyn Huxhold who has done an excellent job of typing the final thesis. And, most especially, I am indebted to my wife, Debra, for her patience in typing the countless preliminary drafts and for her constant encouragement throughout the project.

TABLE OF CONTENTS

Chapters		Page
1	INTRODUCTION	1
	1.1 Calculators - Bridging the Gap	1
	1.2 Outline of the Problem	3
	1.3 Summary of the Thesis	5
2	THE CALCULATOR PROCESSOR	7
	2.1 Introduction	7
	2.2 Processor Design Objectives	7
	2.3 Microprocessors	8
	2.3.1 Algorithm Implementation	9
	2.3.2 Circuit Implementation	12
	2.4 Calculator Element	14
	2.5 Comparison	18
3	THE CALCULATOR	20
	3.1 Introduction	20
	3.2 Functional Description	20
	3.3 The Calculator Circuit	23
	3.4 Stand-Alone Calculator	26
4	SYSTEM DESIGN	28
	4.1 Types of Data Paths	28
	4.2 Control of Data Paths	32
	4.3 Communication Synchronization	37
	4.4 Summary of System Design	41
5	IMPLEMENTATION	44
	5.1 The Calculator Processor	47

Chapters Page

5.2 Display, Drivers, and Display Selector - - - - - - - - 47

5.3 Key Input Decoder - - - - - - - - - - - - - - - - - - 47

5.4 Keyboard and Key Output Control - - - - - - - - - - - 50

5.5 Display Input from Computer - - - - - - - - - - - - - 52

5.6 CP Result and Key Output - - - - - - - - - - - - - - - 52

5.7 Control Registers and Status Lights - - - - - - - - - 55

5.8 Transmitter and Receiver - - - - - - - - - - - - - - - 57

6 OPERATION - 59

6.1 The Monitor - 59

6.2 The Program - 62

7 SUMMARY AND CONCLUSION - - - - - - - - - - - - - - - - - 64

7.1 Brief Review - 64

7.2 Effectiveness of Design - - - - - - - - - - - - - - - 65

7.3 Design Dependency upon CP and Computer - - - - - - - - 65

7.4 Performance Enhancement - - - - - - - - - - - - - - - 67

7.4.1 Speed Considerations - - - - - - - - - - - - - 67

7.4.2 Simpler Design - - - - - - - - - - - - - - - - 69

7.4.3 Increased Capabilities - - - - - - - - - - - - 69

8 REFERENCES - 71

9 APPENDIX - 72

CHAPTER 1

INTRODUCTION

1.1 Calculators - Bridging the Gap

Until the introduction of hand held calculators, engineers have depended heavily on general purpose computers and slide rules to assist them in making day-to-day calculations. These machines have been used to solve a wide variety of problems with different speed and accuracy requirements.

Perhaps most familiar to the engineer has been the slide rule with its numerous scales and moving parts. This analog device is capable of providing results with two or three significant digits of accuracy. It is portable, fairly easy to use and, most important, inexpensive.

Problems of a more complex nature, or those requiring more accuracy, have forced the engineer to rely on the computer. Unlike slide rules, these machines can be programmed to execute a sequence of instructions and subroutines - a highly desired feature. However, the user is still required to write the code, debug the program, furnish the data, and wait his turn to use the machine. Some of these problems were reduced when time-sharing systems brought computer terminals to the user's office.

As new developments in technology were made, a miniature "pocket" computer or calculator replaced slide rules as the engineer's most valuable companion. These machines offered speed and accuracy approaching that of a computer, but at the size and convenience rendered previously only by slide rules. High order functions (e.g., sin, exp) can be computed with eight or more significant digits in a fraction of a second. The problem format used with calculators is also an important characteristic since expressions are entered in a straightforward logical manner. Prices were obviously higher than for slide rules, but not so much that the devices became unaffordable.

The increased computational capabilities afforded by the calculators, while superior to the slide rule, were greatly inferior to the immense powers of the computer. Indeed the calculators did not replace the computers.

Some of these difficulties were overcome with the development of programmable calculators. In addition to the standard calculator features, these machines enabled the operator to specify sequences of instructions. Hence, the machines became a good compromise between large powerful computers and small portable calculators. Unfortunately, the compromise also included an increase in price and size over the pocket machines. The effect of this is that only desk top programmable calculators, costing well over one thousand dollars, are in use today. Furthermore, most applications of the machines do not make full use of the programming features. The engineer then finds himself paying higher costs with less convenience for calculating capabilities which are not fully used.

In recent months it became apparent that it would be practical to design a machine which provided the convenience of a portable calculator and the power of a computer. The primary objectives of such a device would be low cost and portability. Redesigning a computer or a programmable calculator to meet these objectives was considered slightly unrealistic; however, an interface, linking a portable calculator to a computer could become a fair compromise. It was felt that an interface of this type would be valuable for two reasons. First, the operator would be provided with a portable calculator and all the associated conveniences. Secondly, at the operator's discretion, the calculator could be connected temporarily to the computer, tapping its power for solving highly complex problems. At a minimum the user is provided with calculating capabilities on a portable inexpensive machine; more sophisticated problems or those requiring sequences of calculations could be solved

via the computer interface. A remote terminal such as this could also be used to write programs in the machine, access data files, or run programs.

The topic of this thesis is to design and construct a calculator terminal as just described. The machine is capable of operating in a stand-alone configuration or can be linked to a computer to run prewritten subroutines.

1.2 Outline of the Problem

Two separate systems can be identified in this paper: a computer, and a calculator. Each system has its own input channels, output channels, control circuits, and processor. Consequently each machine is a distinct computing entity which can operate quite independently in the absence of the other; no permanent interface is required.

For the purposes of this paper, the computer is defined as a computational machine composed of processor(s), memory, and I/O channels. It is permanently located in a data processing facility and is not accessible for modifications or long term usage.

A calculator is a special purpose machine used for solving simple arithmetic expressions involving the four basic operations (+, -, *, ÷) and possibly higher order functions, e.g. trigonometric or logarithmic. The machine is quite portable, usually pocket sized, and normally is battery powered.

Up to this point each machine has been described independently, each with its own powers and limitations. The problem presented now is how the two systems can be linked together to perform useful computations. The resulting machine, referred to as a calculator terminal, has characteristics which are unique to both machines independently.

The most important property of the calculator terminal is that the operator can select the mode of the machine. The operator would select the

calculator mode for solving problems involving computation powers within the capability of the calculator. Problems of a more complex nature, or those requiring sequences of execution would be done in the computer mode of operation. The function of the machine is highly dependent upon the specific mode of operation. Hence, the characteristics of the calculator terminal will be examined for each mode separately before the design constraints are specified.

When operating as a calculator, the device should provide the operator with a sufficient set of instructions to provide solutions of simple problems. The four basic arithmetic operations: add, subtract, multiply, and divide, are essential members of the instruction set. Higher ordered functions, such as trigonometric and logarithmic, are highly desirable although not mandatory. In general, the larger the instruction set the more convenient the device will be to use and the less often the operator must depend upon a computer.

The physical conveniences afforded by pocket calculators have made these devices much more practical for people to use on a day-to-day basis. A calculator terminal would become nothing more than a permanent office fixture if these conveniences could not be realized. The calculator terminal, as presented here, will not be pocket sized. Clearly, this is due to the high cost associated with producing a portable version of the prototype.

The operator may choose the computer mode to assist him in executing larger, more complicated, problems. The exact amount of processing that the computer will perform depends totally upon the operator's program. As mentioned earlier, it is desired to store prewritten subroutines in the computer and to communicate with them from the calculator terminal.

The operator initially calls the computer into action and specifies which subroutine is to be executed. All pertinent data required to successfully execute the subroutine is entered at the calculator terminal. Once the data has been entered the computer begins execution and no further

responses are required. Upon completion of the subroutine, the computer will indicate the results on the calculator terminal's display.

These characteristics require that the user's subroutine be loaded into the computer prior to being called upon at the calculator terminal. Once this has been done it is not necessary to physically control the computer. That is, all control commands are given through the calculator terminal.

The fundamental characteristics of the calculator terminal can be summarized as follows:

The device is operational as a stand-alone calculator which provides the operator with a sufficient set of instructions to facilitate the evaluation of simple arithmetic expressions. At the discretion of the operator, the calculator terminal can be connected to the computer for the purpose of executing subroutines which are resident in the computer. Control information is transmitted to the computer through the calculator terminal; hence, physical control of the computer is not necessary.

1.3 Summary of the Thesis

This thesis describes an investigation into the design of a calculator to computer interface. The main objective of this study is to improve the capability of calculators by showing that it is practical to interface these devices to a computer. This work relies heavily upon recent advancements in integrated circuit technology.

Chapter 2 examines two processing elements for possible use as the CP of the calculator terminal. An initial design is proposed for each type, so that estimates may be made of the package count and costs. Based upon these figures, a comparison is made and the CP is chosen.

Chapter 3 describes the capabilities of the calculator terminal in the stand-alone mode. The number of available operations is discussed in this

chapter as well as operating procedures. The chapter concludes with a brief look at the electronic circuits of a simple calculator.

The calculator terminal is first presented in Chapter 4 with a discussion of the design criteria. Various data paths are examined for possible inclusion into the final design. The result of this chapter is a functional block diagram of the calculator terminal.

The implementation of the block diagram is discussed in Chapter 5 where a complete set of logic diagrams is presented. The use of the various control signals and how they are synchronized with the CP are also discussed.

The computer monitor is presented in Chapter 6 to give the reader an indication of the power of the computer/calculator terminal pair. Two types of user programs are also presented and discussed.

The thesis concludes with a discussion of the effectiveness of the design. Possible modifications are also presented which may enhance the speed and capabilities of the calculator terminal.

CHAPTER 2

THE CALCULATOR PROCESSOR

2.1 Introduction

The central unit of the calculator terminal is the calculator processing element (CP). The CP is responsible for the operation of the machine in the stand-alone mode. It must be capable of performing simple arithmetic operations and, at the same time, of being easily interfaced to the computer. Two different processing elements are examined in this chapter for this application: a microprocessor and a commercial calculator element.

The examination begins with a discussion of the characteristics of the CP, these can be classified as essential and desirable. It is observed that several of these characteristics can be used to evaluate the effectiveness of the type of processors.

The machines are analyzed separately and methods are proposed for implementing a complete calculator terminal for each. The result of this is a listing of the package count and costs associated with each method. These figures are compared and a particular processor is suggested.

2.2 Processor Design Objectives

The calculator processor must be designed so that the user is provided with a sufficient set of operations to enable the evaluation of simple arithmetic expressions in a stand-alone mode. Furthermore, the device must be easily interfaced to a computer when problems of a more complex nature are involved. Essential characteristics of the processor will include:

1 - Problem format similar to that of a calculator.
2 - Basic arithmetic operations (+, -, *, ÷).
3 - Inexpensive - low cost, low package count.
4 - 8/10 decimal digits of accuracy.
5 - Easily interfaced.

The following parameters, while not essential, are considered to be highly desirable:

1 - Low power consumption, small number of power supplies.
2 - Higher order functions or operations (trigonometric, logorithmic).
3 - Moderate speed.
4 - Small physical size.

The problem format of the processor should be similar to that of a calculator. Problem entry must be easily understandable and logically organized. The problem entry format will be:

⟨expression⟩ = ⟨operand⟩ ⟨binary operator⟩ ⟨operand⟩ =
or ⟨operand⟩ ⟨unary operator⟩

where,

operand = expression, number
binary operator = (+, -, *, ÷, ...)
unary operator = (SIN, LOG, 1/x, ...)

These operations should be accurate to eight or ten decimal positions and be completed in a reasonable amount of time. Obvious restrictions are imposed on the physical size and cost of the stand-alone device.

For purposes of comparison, it will be convenient to estimate several parameters for each processor type. Three items can be categorized for this purpose:

1 - Package count based on a preliminary design.
2 - Cost of each type based on current market figures.
3 - Number of operations realizable at the above cost and package count.

2.3 Microprocessors

One of the most powerful devices currently on the market is the microprocessor. This device normally consists of one or two integrated circuit packages which may, with the addition of external memory and I/O, be programmed to meet almost any design objective. Although each microprocessor has its own unique instruction set and I/O requirements, it is felt that enough similarities do exist to discuss the devices in general terms. The problems associated with using a microprocessor as the CP can be divided into two

groups: interface requirements and algorithm implementation. The following sections describe how a microprocessor could be used to implement the calculator terminal.

2.3.1 Algorithm Implementation

Microprocessors have an instruction set which usually consists of simple one or two machine cycle commands. These can be divided into four classes: data transfers, arithmetic, control and I/O. Each instruction causes a different action to be taken by the processor.

Data transfer instructions are used to move information from one storage location to another. All microprocessors enable data transfers between memory and registers, and register and I/O devices. In addition, several of the more sophisticated processors have indirect addressing capabilities.

Arithmetic instructions are usually binary and may operate on data in memory or in registers. Typically, these instructions include: add, subtract, shift, and logical operations. Instructions of a more complex nature are normally not found in microprocessors and must be programmed.

Utilizing a microprocessor as the processing element of the calculator will require the use of algorithms which will accept input data, perform the required operations, and present results to the operator. The algorithms are implemented as a program which lists machine instructions necessary to perform these functions. The programs are permanently stored in a read-only memory (ROM) contained in the calculator. It will also be necessary to store temporary data, input files, and intermediate results - a random-access memory (RAM) can provide these capabilities.

The program could consist of a system monitor and several subroutines. The monitor is used to coordiante processing requirements by controlling input/output and subroutine execution functions. During the normal input stream,

the task of the monitor will be to accept numeric data and store them as operands. An arithmetic operation could then be decoded as an address of the relevant subroutine. A subroutine exists in the program which is capable of evaluating each instruction the user may specify. In this manner the size of the monitor is kept to a minimum and program modularity is retained. System expansion can also be easily accommodated by adding ROM's. A typical monitor could be designed as shown in Figure 2.1.

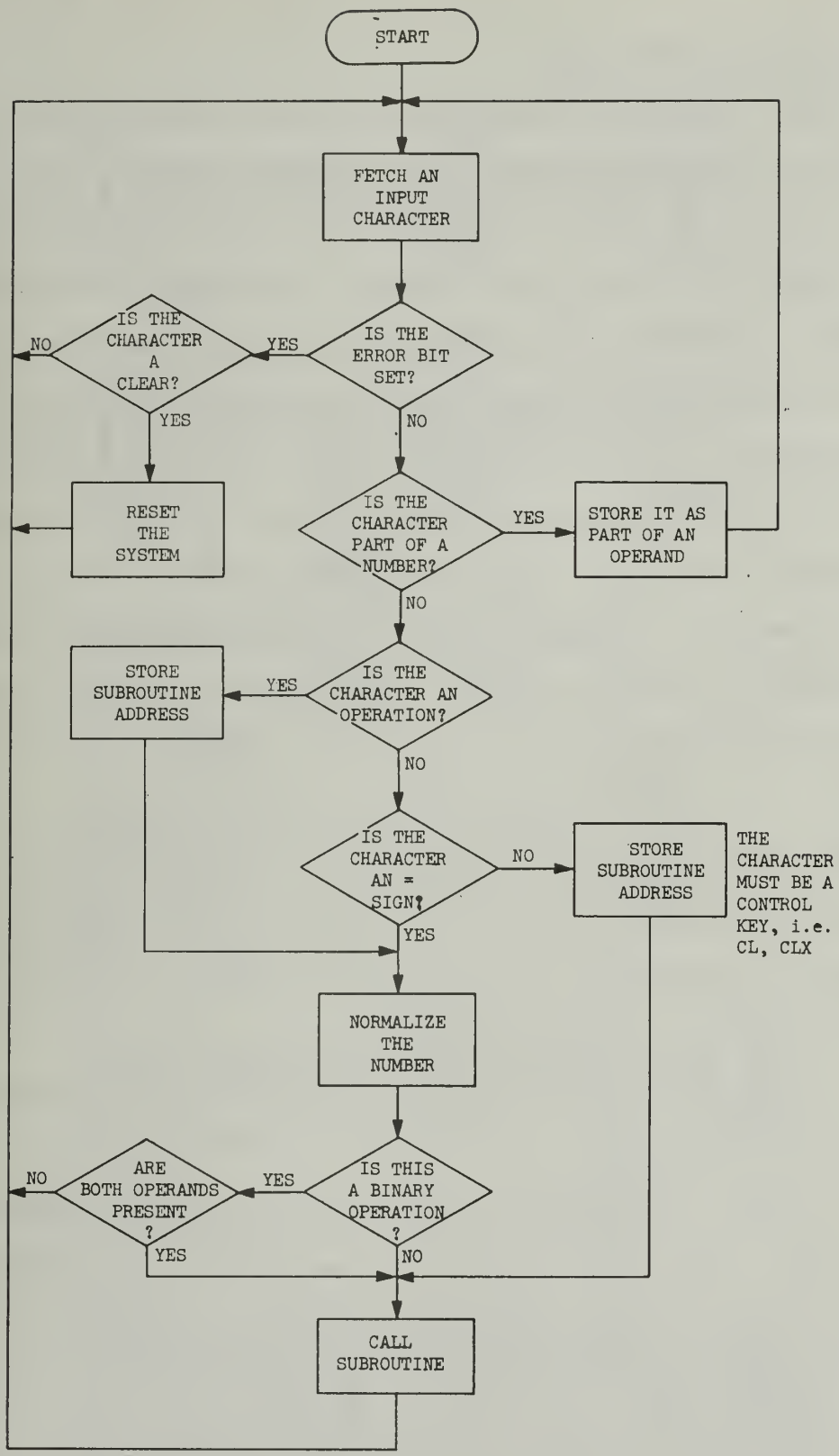
The length of the program and the amount of read/write storage required will depend upon the instruction set of the processor and the number of arithmetic operations that are to be implemented. An estimation can be given for the amount of ROM and RAM needed to implement the monitor and the four basic operations. Separate estimates can then be provided for memory requirements needed to implement higher order functions on an individual basis. These figures are indicated in Table 2.1, assuming a microprocessor such as the Intel 8080.

FUNCTION	ROM	RAM*	PACKAGES
+, -, x, ÷, CLR	1k x 8	256 x 8	1 pcs - 1k ROM 1 pcs - 256 RAM
+, -, ÷, CLR $\sqrt{x}$, x^2, 1/x, π, e^x	2k x 8	256 x 8	1 pcs - 2k ROM 1 pcs - 256 RAM
+, -, x, ÷, CLR $\sqrt{x}$, x^2, 1/x, π, e^x SIN, COS, TAN, ARC	3k x 8	256 x 8	1 pcs - 2k ROM 1 pcs - 1k ROM 1 pcs - 256 RAM
+, -, x, ÷, CLR $\sqrt{x}$, x^2, 1/x, π, e^x LOG, LN, Y^x SIN, COS, TAN, ARC	4k x 8	256 x 8	1 pcs - 4k ROM 1 pcs - 256 RAM

*Smallest amount available.

Memory Estimates for a Calculator Terminal Design Utilizing a Microprocessor.

Table 2.1



Proposed Software Monitor for a Calculator Terminal Employing a Microprocessor as the CP.

Figure 2.1

2.3.2 Circuit Implementation

Circuit implementation of a microprocessor deals with the problems of connecting the various integrated circuit packages together to form a useful system. The problems can be divided into four areas: processor, processor control, memory, and I/O devices. Each area will be discussed separately at this time so that the cost and package count of the system can be estimated.

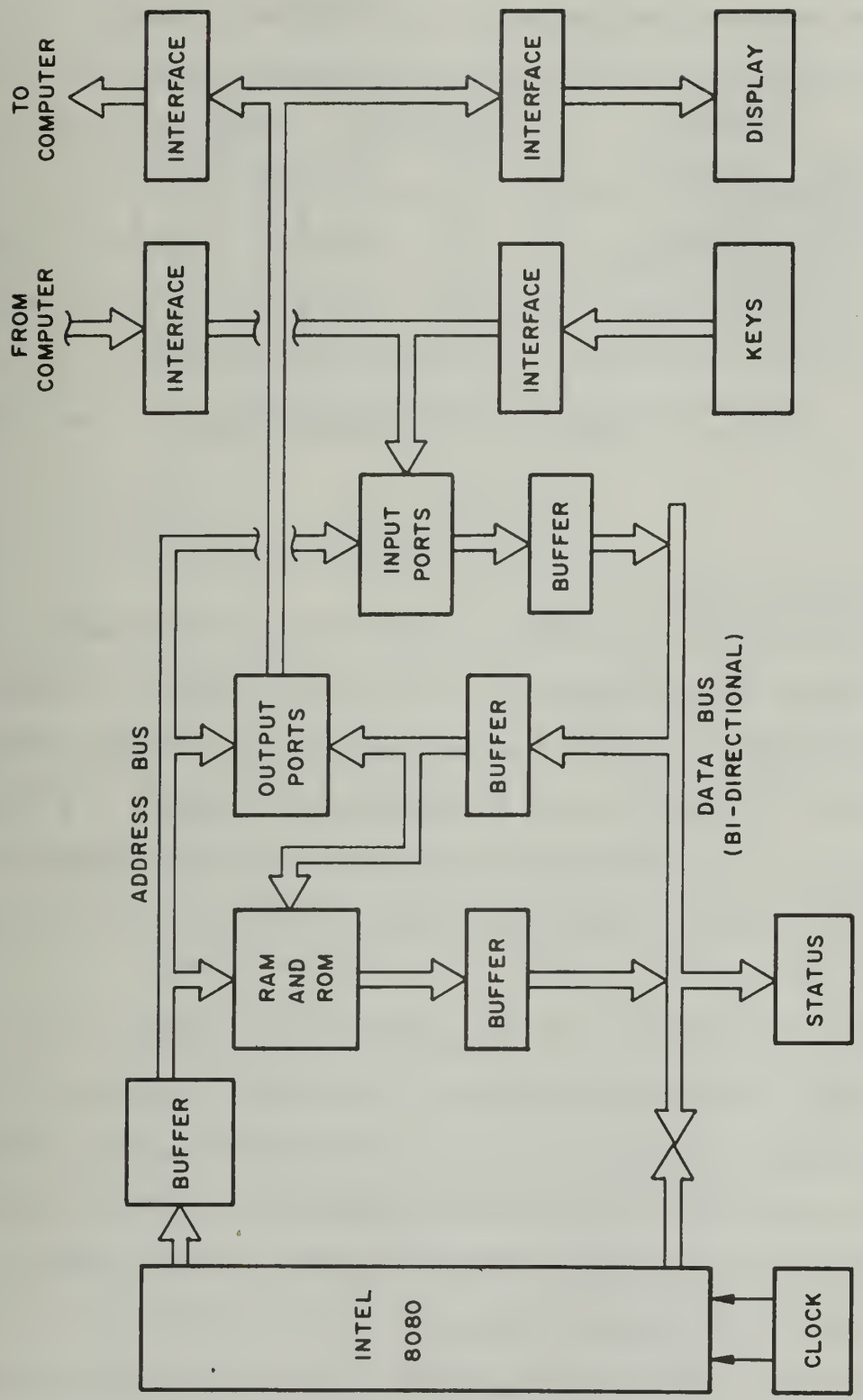
The processor and processor control portion of the system are used to accept input data, execute programs, and output the results. The network consists of the microprocessor and clock generator. One package contains the microprocessor while six or more are required to generate the clock pulses.

The amount of ROM and RAM used in the memory depends upon the length of the programs. However, regardless of the amount, a package count in the neighborhood of four to six will be required to interface the memory to the processor.

Each I/O port of the microprocessor can be assigned an address to facilitate data transfers. Three I/O ports will be required to connect the keys, display, and computer to the processor. At least four or six packages will be required for each I/O interface.

A generalized system is given in Figure 2.2 which illustrates how the system might be implemented.

In summary, realizing a calculator terminal with a microprocessor can be easily done. The computer is treated as an I/O device to facilitate operation in the computer mode. Implementing arithmetic operations on a microprocessor becomes quite involved since programs must be written and stored in ROM. Many parameters of the calculator depend upon the program, but any amount of accuracy can be obtained by using the correct algorithms. Package count and cost for a calculator terminal, capable of only the basic arithmetic



Block Diagram of a Calculator Terminal Employing a Microprocessor as the CP.

Figure 2.2

operations, are given in Table 2.2. Using these figures, a minimum system would include at least 38 packages at a cost of $270.00 or more.

Component	Packages	Cost
Microprocessor	1	$120 - 360
ROM (1k x 8)	1	$ 60 - 100
RAM (256 x 8)	1	$ 25 - 50
Processor Control	6	$ 10 - 20
Memory Interface	6	$ 30 - 50
Device Interface	18	$ 30 - 75

Estimates of Package Count and Cost of a Calculator Terminal Utilizing a Microprocessor.

Table 2.2

2.4 Calculator Element

Several manufacturers market a complete calculator processor for use in pocket versions of the machines. The processor is normally contained in one or two packages and operates from two power supplies. Major problems are encountered when an attempt is made to interface the processor to a computer. In the following section, the characteristics of a calculator element are presented. This leads to a discussion of how the processor can be used as the CP.

A calculator element is an integrated circuit, designed for the sole purpose of being used as the processor of a stand-alone calculator. The number of arithmetic operations which may be evaluated depends upon the particular unit chosen; some are capable of evaluating only the four basic operations (+, -, *, ÷) while others allow more complex functions. All units have two I/O ports: a keyboard and a display.

The display consists of n digits which are driven by the calculator with a set of output (segment) lines. Display data is normally available as a 7-segment code in a digit serial manner. The calculator is also

provided with a set of input signals which are generated by a keyboard matrix and used to specify numeric entries and arithmetic operations. Normally, the calculator element contains logic for debouncing the switches and blocking out multiple key pushes.

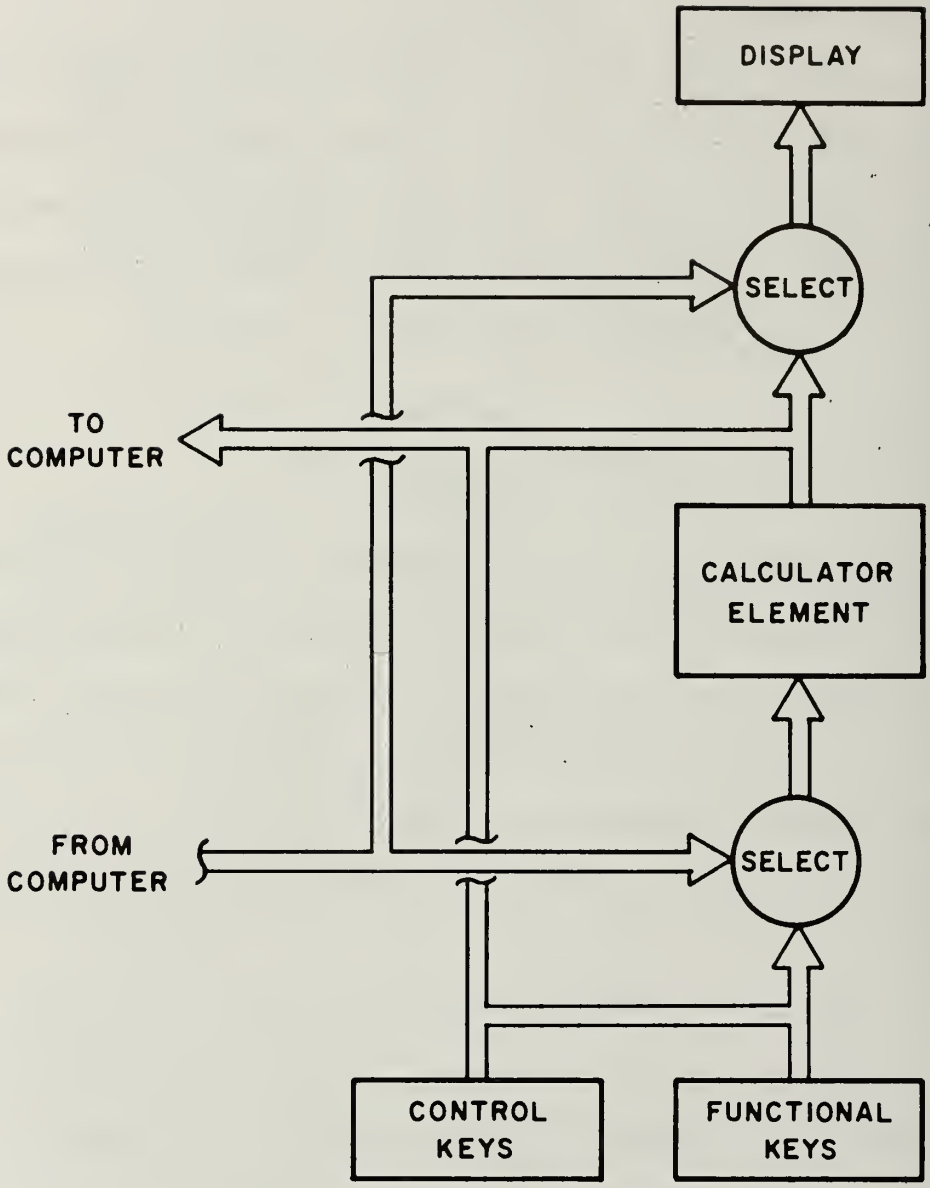
Several problems arise when an attempt is made to interface with the calculator element. The keyboard and the display must be made accessible to the computer. At the same time, the computer must be able to enter information into the calculator and obtain results. These problems can be divided into four areas for easy discussion:

1 - Keyboard
2 - Display
3 - Computer to calculator transfers
4 - Calculator to computer transfers

The functional keys which are required by the calculator element must be readable by the computer and the calculator when operating in the computer mode and calculator mode respectively. In addition to these functional keys, the computer will require a few control keys. Data "switches" can be used to direct the flow of key information as shown in Figure 2.3.

The display should be capable of representing numeric output from either the computer or the calculator. This can be accomplished by having two sets of segment lines available, one set generated by the computer and the other set generated by the calculator. The display can then be switched from one to the other as desired. Display data from the computer can be stored in a memory device within the calculator terminal.

Transfers of information from the computer to the calculator must be entered in the form of key pushes. This can be done by adding a new keyboard to the system for use by the computer. This keyboard, which is functionally equivalent to the user keyboard, could be controlled by signals from the computer.



An Elementary Calculator Terminal Design
Employing a Calculator Element as the CP.

Figure 2.3

Calculator results which are to be transmitted to the computer must be converted from 7-segment code to binary-coded-decimal (BCD). The result can be stored in a RAM and transmitted as a whole, or the digits can be transmitted serially as they appear on the segment lines.

The four problems discussed will require a rather sophisticated control circuit to monitor the state of the various switches as Figure 2.3 indicates.

The calculator terminal can be realized by using a calculator element as the CP. The number of operations which can be evaluated depends upon the specific calculator element chosen. Interfacing this device to a computer is quite complicated and requires the use of at least 5 or 6 additional circuits. Table 2.3 lists estimates for the package count and cost of the calculator terminal using a calculator element as the CP. Such a system will probably contain between 80 and 120 integrated circuit packages at a cost ranging from $50 to $150. Various calculator elements are listed in Table 2.4 showing the types of available arithmetic operations.

Device	Packages	Cost
Calculator Element	1 or 2	$10 - 75
Key circuit	20 - 30	10 - 20
Display circuit	20 - 30	10 - 20
Calc → Comp Circuit	20 - 30	10 - 20
Comp → Calc Circuit	20 - 30	10 - 20

Estimates of Package Count and Cost for a Calculator Terminal Design Utilizing a Calculator Element.

Table 2.3

Mfg.	Type	# Digits	Decimal Point	Functions
Mostek	MK 5013 MK 5014	12	floating	+, -, *, ÷, M
Mostek	MK 5010P	10	floating	+, -, *, ÷
Mostek	MK 5012P	10	floating	+, -, *, ÷
Texas Inst.	TMS 0100	8 or 10	floating	+, -, *, ÷
Texas Inst.	TMS 0117	10	floating	+, -, *, ÷, INC, DEC
AMI	S 2144	8	floating	+, -, *, ÷
MOS Tech.	MPS 2525 MPS 2526	10+ 2 exp	scientific notation	+, -, *, ÷, Y^x $\sqrt{x}$, x^2, 1/x, M, π, e^x ARC, SIN, COS, TAN, LN, LOG

Examples of Commercial Calculator Elements and Their Characteristics.

Table 2.4

2.5 Comparison

The discussions in this chapter have examined the possibility of using microprocessors and calculator elements as the CP of the calcaulator terminal. Several problem areas encountered in both processor types have been discussed and tentative solutions have been given. At the conclusion of each investigation, an estimate was given of the package count and cost associated with the design. Table 2.5 summarizes these estimates and enables a comparison to be easily made.

	Operations	Package Count	Cost
Microprocessor	+, -, *, ÷	40	$270
Microprocessor	all	40	$450
Calculator	+, -, *, ÷	80 - 100	$50 - 70
Calculator	all	80 - 120	$70 - 150

Comparison of Package Count and Costs for a Calculator Terminal Design Utilizing a Microprocessor or a Calculator Element.

Table 2.5

Three conclusions can be made from the data in Table 2.5. With regard to the microprocessor design it is apparent that the package count is not directly related to the complexity of the operation set. It is obvious that microprocessor implementations cost about five times more than calculator element implementations. However, microprocessors can be designed with less than half the number of packages needed in a calculator element design.

The initial calculator processor requirements specified a low cost, portable device which is capable of solving simple arithmetic problems. Clearly, both of the processor types can meet the last requirement. It is only in choosing a low cost processing element, which can also be made portable, that a conflict arises. Since it is only intended to demonstrate the feasibility of such a device, it was decided to relax the portability requirements and use the least expensive processor type - the calculator element. Indeed, an IC manufacturer might conclude that a microprocessor design would be the least expensive for high volume production. Hence, the decision is somewhat biased since only a prototype has been constructed at this time. Referring to Table 2.4 and 2.5 it is noted that a design using a highly sophisticated calculator element can be realized as easily as one using a less powerful element. The processing element which was chosen for the calculator terminal is the MOS Technology MPS 2525/2526 calculator array.

CHAPTER 3

THE CALCULATOR

3.1 Introduction

The processing unit used in the calculator terminal is the MOS Technology MPS 2525/2526 calculator array. The CP consists of two 28-pin DIP packages utilizing low threshold voltage PMOS technology. The unit does not require an external clock and operates from two power supplies.

Several topics will be discussed in this chapter pertaining to the operation and implementation of a stand-alone calculator using the MPS 2525/2526 processor. A brief description of the mathematical operations and problem format will be presented to clarify the processing capabilities of the calculator. Also considered is the circuit design of a simple stand-alone calculator using only a keyboard, display, and processor.

3.2 Functional Description

The calculator operates in a scientific algebraic format and is capable of handling operands with an absolute value between 1×10^{-99} and $(10 - 10^{-9}) \times 10^{99}$. Entry numbers or result data falling outside this range cause an overflow, which results in an error message on the display.

The display consists of fourteen digits which are used to indicate currently entered numbers or computational results. Included in the display are a mantissa sign, ten digits of mantissa, an exponent sign, and two digits of exponent in the following format:

MAN SIGN	MAN 9	MAN 8	MAN 7	MAN 6	MAN 5	MAN 4	MAN 3	MAN 2	MAN 1	MAN 0	EXP SIGN	EXP 1	EXP 0

Display Format of the Calculator Terminal.

Figure 3.1

Input numbers are displayed exactly as the keys are depressed. Result data is represented in floating point notation (exponent of 0) if possible, otherwise the representation consists of a mantissa between 1.0 and 9.999999999 in conjunction with an exponent. A special character, Γ, appears as the mantissa sign in the event an overflow or illegal operation occurs.

Data entry is accomplished in one, or possibly two phases. The first such phase occurs when entering the mantissa. The first ten digit key depressions are entered into the calculator as the mantissa, the eleventh and successive digit entries are ignored. The decimal point /·/ and change sign /CHS/ keys may be entered at any point during the mantissa input stream. The second phase is optional and only occurs when the enter exponent key /EEX/ is depressed. During this phase two digits of exponent may be specified. A /CHS/ entry while in this mode will negate the sign of the exponent. At any point in either of the two phases a clear entry key /CE/ will void the number currently being entered. Other data stored in the unit is not affected by this action.

The CP contains an x and a y register which are used to store operands. The x register is permanently connected to the display and is used to hold operands during entry and result data upon completion of execution. The y register is not directly accessible by the operator and must be transferred to the x register for manipulation.

The instruction set of the calculator processor can be divided into four classes: double operand, single operand, control, and miscellaneous as listed in Table 3.1.

Double Operand	Single Operand	Control	Misc.
+	SIN, ASIN	=	M
-	COS, ACOS	CL, CE	π
*	TAN, ATAN	(,)	x:y
÷	LOG, ALOG	o/R	
Y^x	LN, ALN		
	1/x, $\sqrt{x}$, x^2, e^x		

Operation Set of the CP.

Table 3.1

Double operand instructions include +, -, *, ÷, and Y^x. Expressions may contain one or more such instructions in the following format:

⟨number⟩ ⟨doi⟩ ⟨number⟩ ... ⟨number⟩ ⟨=⟩

where doi indicates a double operand instruction. Execution begins when either the equal sign is entered or a second double operand instruction is specified. In either case the x and y registers contain the same result upon completion.

Single operand instructions include SIN, COS, TAN, LOG, LN, 1/x, $\sqrt{x}$, e^x, and x^2. The inverse functions of the trigonometric and logarithmic instructions are also available. The instructions operate on the number contained in the x register under the following format:

⟨number⟩ ⟨soi⟩

where soi indicates a single operand instruction. Execution begins immediately after entering the instruction. The result is placed in the x register and the y register is unaffected.

Equations containing a mixture of single and double operand instructions may be evaluated:

a [-] b [SIN] [+] c [$\sqrt{x}$] [=]

produces:

a - SIN b + $\sqrt{c}$

Parentheses may also be included in the expression to force the evaluation of a subexpression. Up to two levels of parentheses nesting are allowed:

/(/ /(/ a /+/ b /)/ /Y^X/ c /)/ /-/ /(/ /(/ d /+/ e /)/ /Y^X/ f /)/ /=/
/$\sqrt{x}$/ /1/x/ /SIN/ /Y^X/ g /=/

derives:

$$\text{SIN}\left[\frac{1}{\sqrt{(a+b)^c-(d+e)^f}}\right]^g$$

The control keys consist of: =, CL, CE, (,), and o/R, which are used to specify to the calculator how a particular problem is to be evaluated. The =, (,and) keys alter the normal sequence of instruction execution as explained earlier. The /CE/ instruction voids a number which is currently being entered as opposed to the /CL/ command which resets the calculator and clears all registers. /CL/ is the only command the machine will accept after an error has been made. The /o/R/ key selects the units of the operands which are used for trigonometric functions.

Three keys, π, M and x:y are provided for entering prespecified data into the x register. The /π/ command deposits the number 3.141592654. The /M/ instruction is used to access or store information into a memory register. The interpretation of this command is context sensitive. Entering the /M/ command in the midst of an expression causes the memory number to be deposited into the x register. Conversely, entering the /M/ command after an equal sign causes the number in the x register to be stored in M. Lastly, the /x:y/ command interchanges the contents of the x and y register.

3.3 The Calculator Circuit

The calculator circuit contains a calculator processor, a display and a keyboard. Using these three elements, a stand-alone calculator can be constructed with the capabilities previously discussed. This section examines

the operation of the fundamental stand-alone calculator. An understanding of these principles is essential in the development of the interface circuitry.

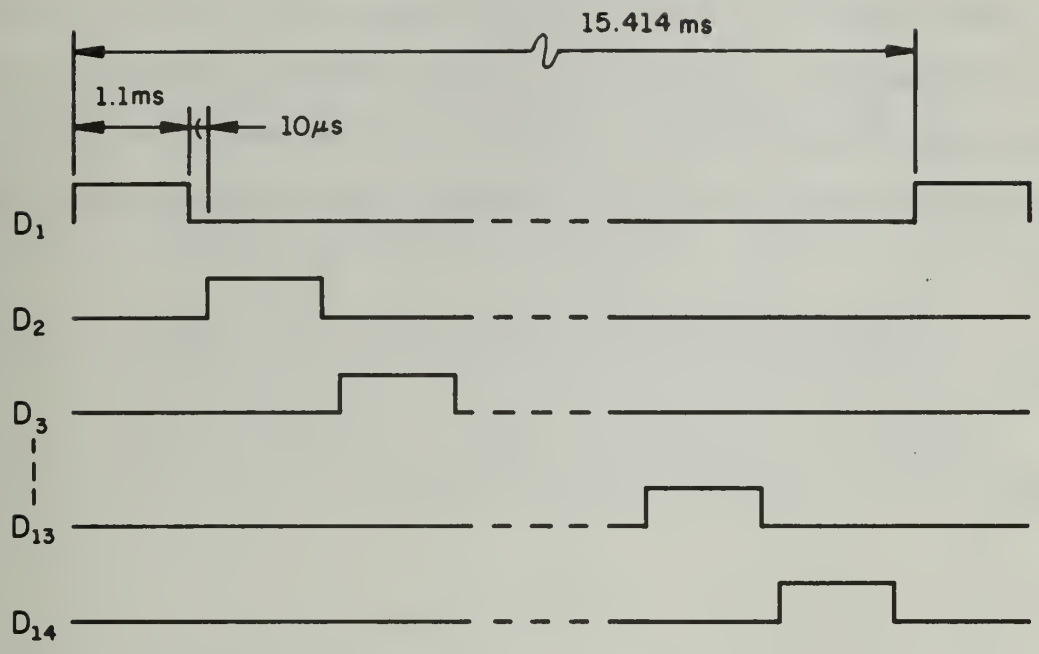
As mentioned earlier, the pin restrictions imposed upon calculator processing elements have made it necessary to perform I/O through a set of multiplexing, segment, and keyboard lines.

The calculator processor of concern here has fourteen multiplexing lines called DIGIT LINES which are referred to as D_1, D_2, D_3 ... D_{14}. Figure 3.2 illustrates the waveforms present on each line and clearly shows that at most one D_n line is active at a time. Each pulse indicates that a particular piece of data can be found on the segment or keyboard lines. In the waveforms shown, a DIGIT TIME is defined as the interval of time between a positive edge on D_n and a positive edge of D_{n+1}, this value is 1140 + 20 = 1160 μs. A CYCLE TIME is the time required for a complete cycle through all fourteen digit lines; 14 x 1160 μs = 16.25 ms. These signals will be used throughout the interface design to synchronize the calculator terminal with the CP.

The segment lines specify which segments of the nth digit are turned on during the nth digit time. The data present on these lines is updated during the 20 μs period between actual digit line pulses. Figure 3.3 illustrates this circuit for one of the fourteen digits of the display.

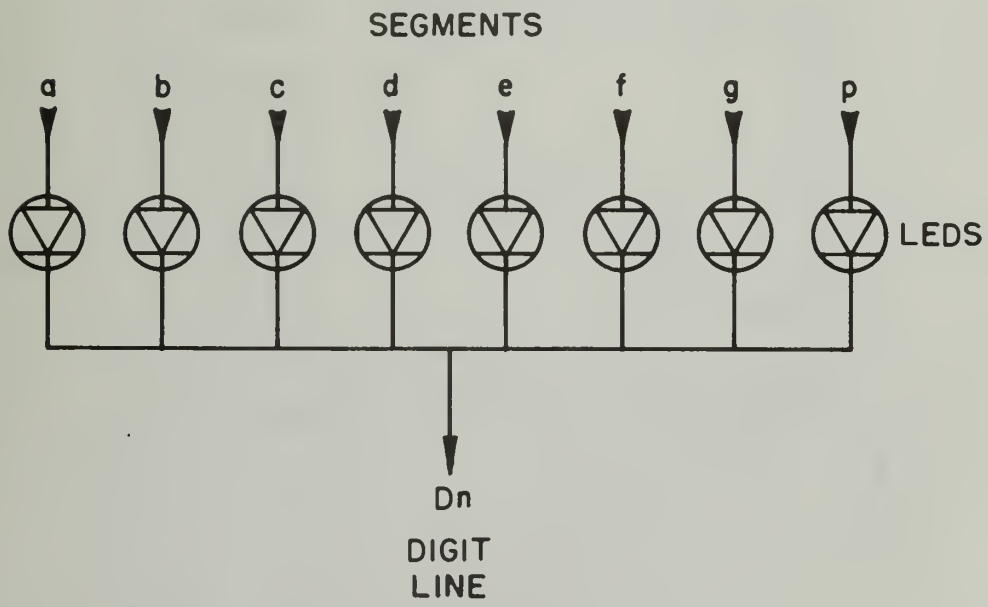
Electrical properties of the digit and segment lines, which will determine the characteristics of the drivers, will be discussed later.

The keys are electrically located at the intersection points of a 14 x 3 matrix. The digit lines form the matrix columns as shown in Figure 3.4. Pushing a key located on the nth column causes the waveform of the nth digit line to appear on one of the row lines. This information is used to determine exactly which key was pushed. The calculator is equipped with circuitry to detect multiple key depressions and contact bounce.



Waveforms Present on the Digit Lines.

Figure 3.2

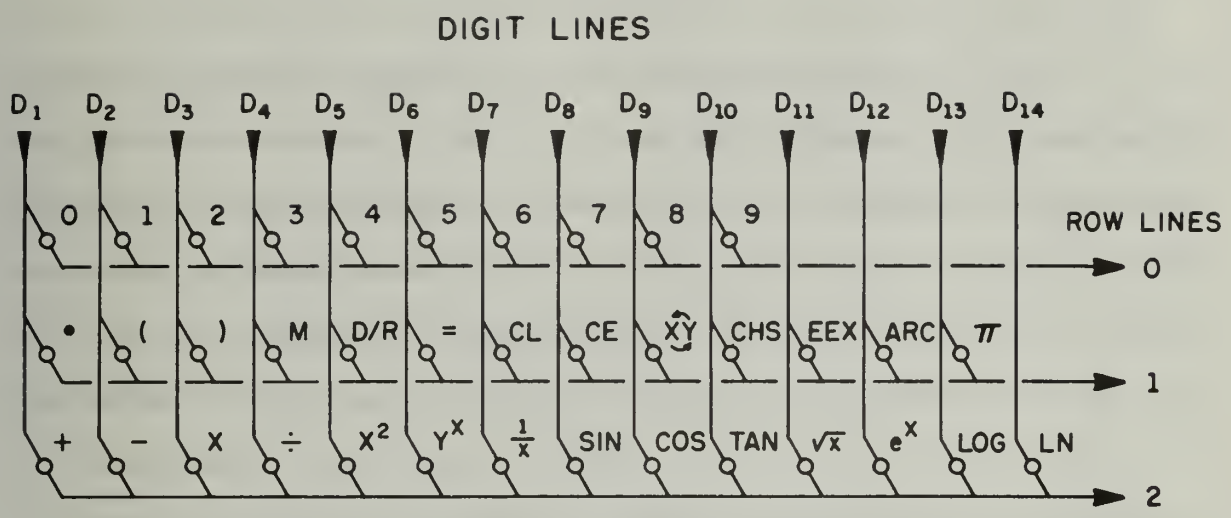


Equivalent 7 - Segment Circuit for One Digit.

Figure 3.3

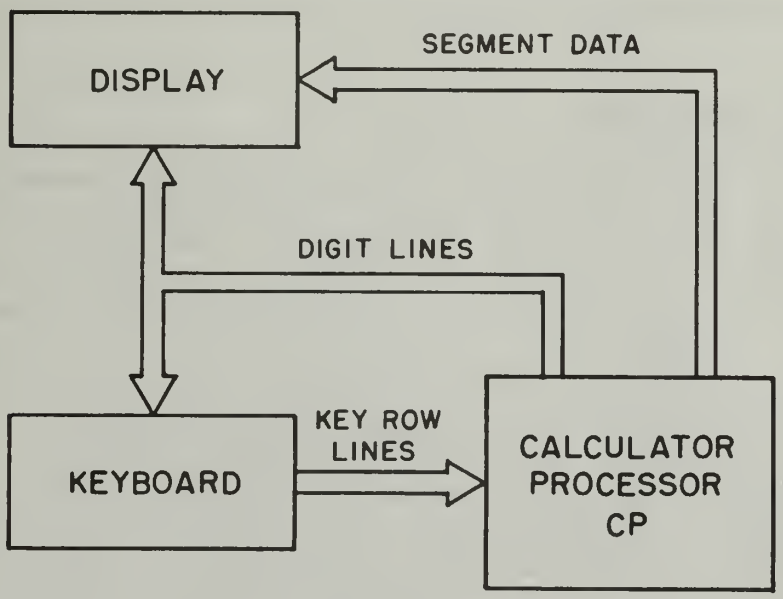
3.4 Stand-Alone Calculator

The calculator, as described in this chapter, provides the operator with a large selection of operations which may be used to solve equations in a stand-alone mode. The circuit is rather simple to realize and consists of only three devices as shown in Figure 3.5. Consideration can now be directed toward designing a circuit to interface the calculator to a computer.



Keyboard Matrix.

Figure 3.4



Block Diagram of a Stand Alone Calculator.

Figure 3.5

CHAPTER 4

SYSTEM DESIGN

It was previously determined that the calculator terminal provides the operator with two computing modes with which to solve a wide variety of problems. The calculator, or stand-alone, mode is effective for evaluating simple mathematical expressions which do not require programming features. Problems which fall outside the capabilities of the calculator terminal may be solved by reconfiguring the system to include a computer. Reconfiguration occurs on command from the operator through a mode selector switch and is accomplished by controlling data transfers between the display, keyboard, CP, and computer.

This chapter describes a method which may be employed to enable a configuration of the system which is compatible with the user selected mode. A definition is first presented of the various types of data paths which the system may require in certain instances.

4.1 Types of Data Paths

Operation of the machine in the stand-alone mode necessitates data transfers between the CP, display, and keyboard. Information is entered into the system when the operator strikes a key and the corresponding key code enters the CP. Information processing is performed in the CP and result data is transmitted to the display and ultimately to the operator.

The CP must be allowed to accept key information and output result data on the display; indicating that a functional keys → CP and CP → display data paths are necessary. When the calculator terminal is operating in the stand-alone mode, the CP is able to use the keys and display as I/O channels. At no time is the CP allowed to reconfigure the system.

Several additional data paths must be included when a computer is introduced into the system. It is necessary for the computer to accept information from the functional keys and output result data on the display. Otherwise, the computer would not have access to the I/O channels. In addition, the computer should also be provided with some means of accessing the CP so that shared processing can occur. This requires that the computer be able to enter data into the CP and obtain results. The four new data paths needed to enable the basic operation of the calculator terminal in the computer mode include: computer → CP, CP → computer, computer → display, and functional keys → computer.

These data paths are indeed fundamental to the computer mode of operation. However, they fail to provide the operator with a mechanism for directly controlling the computer.

During computer interactions the operator will be required to specify subroutine numbers, enter keyboard parameters, and control the execution of programs. All of these actions involve the direct transmission of control information from the operator to the computer. The present calculator input mechanisms would require that the operator enter a response on the functional keyboard, known to both the CP and the computer (e.g. SIN, M, ., +...). Obviously, this will result in ambiguous key interpretations, that is, a key's definition will be a function of the state of the computer. Such a situation is obviously undersirable, a better solution being to add a small number of keys to the calculator terminal for controlling the computer. The set of keys available to the operator then becomes the union of all functional keys and all control keys. A new data path, control keys → computer, transmits the control key information directly to the computer.

The exact number and definition of the control keys should be sufficiently general to enable the efficient control of the computer and

subroutines. Six keys, governing different phases of the computer operation, can provide the operator with the necessary control capability:

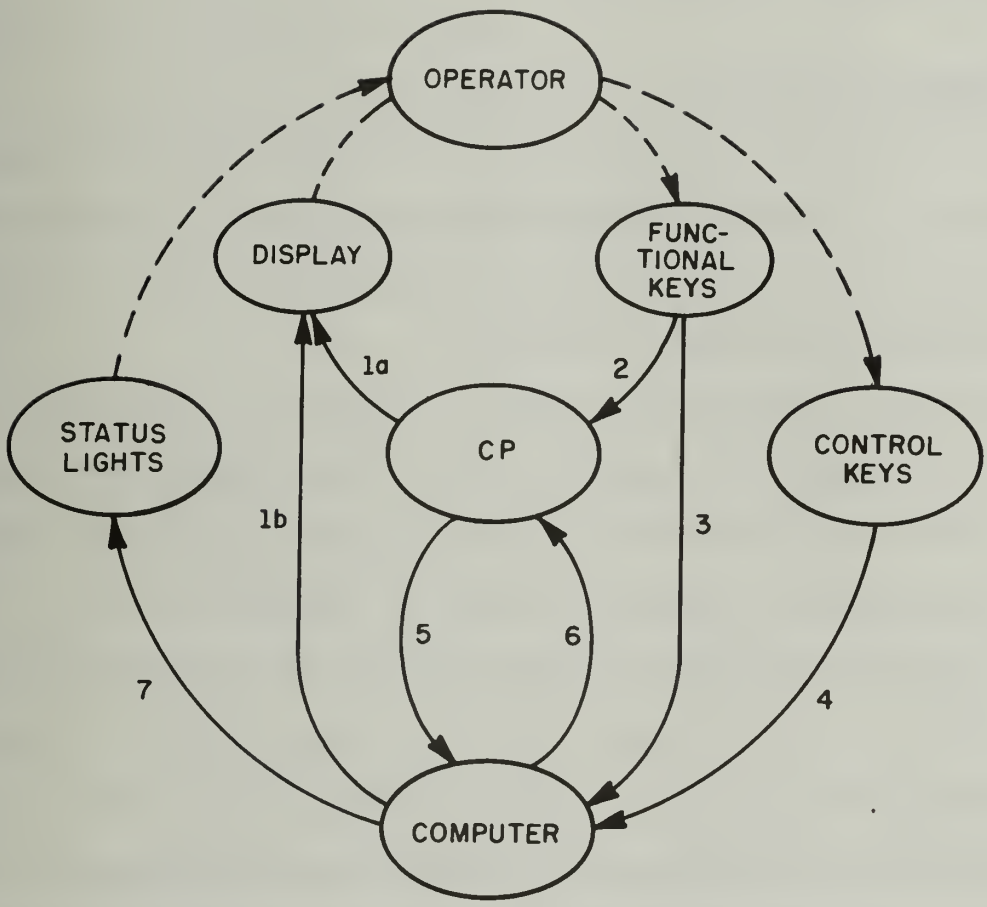
- call computer
- enter subroutine number
- enter subroutine parameters
- output next result number
- abort the subroutine
- halt computer interaction and return to stand-alone calculator mode.

In an analogous manner, the operator will often need to know the state of the computer. Operator responses such as entering subroutine parameters or specifying subroutine numbers are dependent upon this knowledge. Obviously, the display could be used to inform the user of any actions necessary on his part. However, the problem remains that this type of information, when presented on the display, might be confused with numerical result outputs. As in the case of the keyboard, the best solution is to provide a different output mechanism to transmit information of this type.

The method selected employs a set of status lights to indicate the state of the computer and whether or not any action is necessary on the part of the operator. The computer → status light data path is necessary to activate the lights. As before, the definition of the status lights should be sufficiently generalized to indicate the state of the computer under most circumstances. Six lights, indicating the computer's status and requesting operation:

- enter subroutine number
- enter parameter data
- input data error
- execution error
- subroutine finished
- output data valid

Eight types of data paths have been examined and discussed up to this point. The completed system in Figure 4.1 illustrates the three closed-loop information paths between the operator, CP, and computer. The following



Three Closed-Loop Information Paths Made
Available with the Computer/Calculator Terminal Pair.

Figure 4.1

numerical types are assigned to the various data paths to assist in future discussions:

1a - CP → display
1b - computer → display
2 - functional keys → CP
3 - function keys → computer
4 - control keys → computer
5 - CP → computer
6 - computer → CP
7 - computer → status lights

4.2 Control of Data Paths

As the calculator terminal enters the computer mode of operation, the system must reconfigure as determined by the software. It was mentioned earlier that reconfiguration is achieved by controlling the flow of information along the data paths indicated in Figure 4.1.

Data control modules may be placed in a data path to regulate the flow of information. In general, data paths which transmit computer generated information should not be interrupted by a control module. It is only data produced by the operator or the CP which must be regulated. The exact placement of the modules will now be examined as related to each type of data path defined earlier.

Type 1 - Display Data

The type 1 data path pertains to output information, generated by the computer and the CP, which is directed to the display. The CP output data is in the form of an 8-bit segment code and may be transferred directly to the display. Computer output information is BCD encoded and stored in a small RAM within a logic network. The BCD must be 7-segment encoded prior to transmission to the display.

Clearly, result data from the CP (type 1a) is allowed to enter the display when the calculator terminal is operating in the stand-alone mode.

However, either processing element may output data when the machine is under program control. Hence, a data selector module is required to allow one, and only one source of display information.

The subsystem of Figure 4.2 indicates how the display network is designed using an 8-bit data selector module. The display select (DS) control line is provided so that the source of display data may be specified.

Types 2 and 3 - Functional Key Data

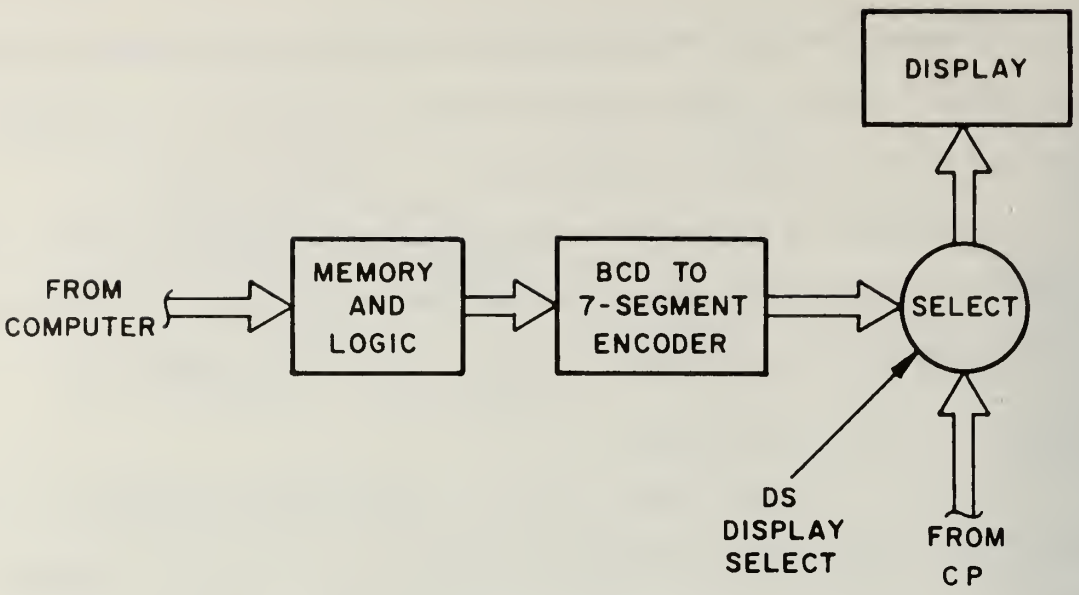
Types 2 and 3 data transfers are used to enter functional key information into the calculator and computer respectively. It will be recalled from Chapter 3 that functional key information which is entered into the CP consists of digit line waveforms present on one of three row lines. Transmitting this information to the computer is slightly more complicated since the column and row location of the key must be encoded. Using a 14 x 3 keyboard matrix implies that a 6-bit key encoding is required before transmission to the computer can occur.

When the calculator terminal is operating in the stand-alone mode it is necessary for the key information to be entered into the CP. When computer interactions are desired the functional key data can be transmitted to the computer, the CP, both, or neither as determined by the software.

The subsystem shown in Figure 4.3 illustrates how the control modules are placed to regulate the flow of functional key information. The enable functional key → CP (EFKC) control line is provided to control the 3-bit key entries into the CP. Similarly, the enable functional key output (EFKO) control line regulates the flow of 6-bit key data to the computer.

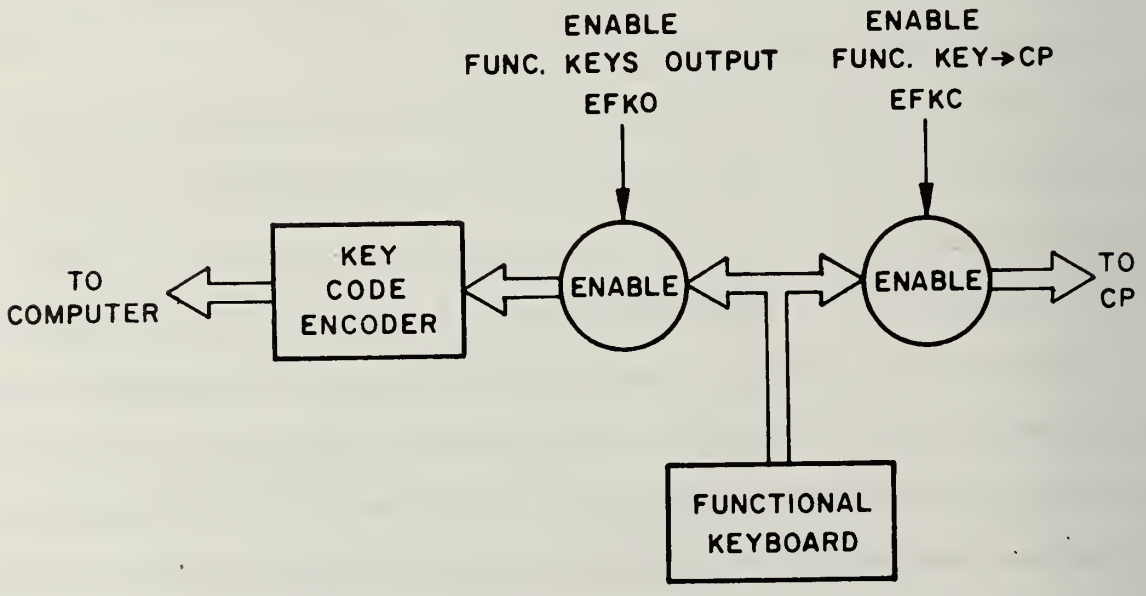
Type 4 - Control Key Data

Type 4 data transfers were defined as the flow of control key information to the computer. It would be convenient to locate these keys on the



Display and Display Selector.

Figure 4.2



Control of Functional Key Information.

Figure 4.3

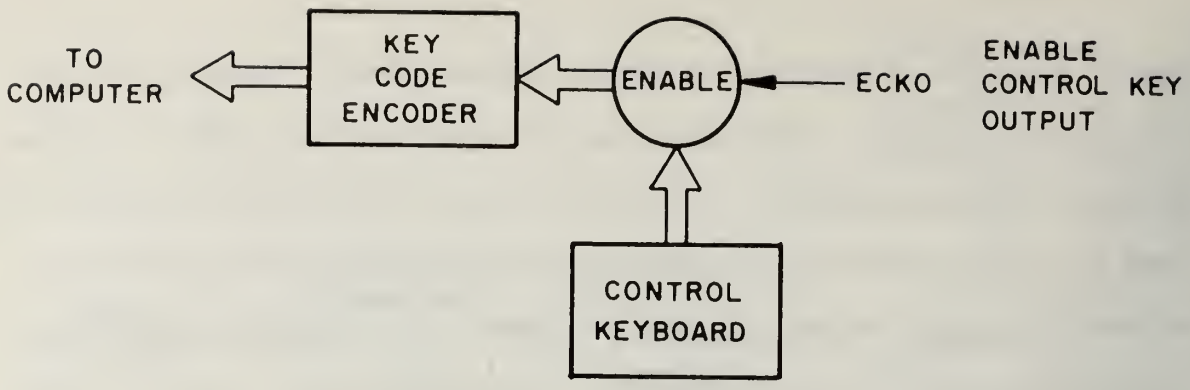
same 14 x 3 keyboard matrix used for the function keys. Figure 3.4 shows that several locations on the matrix may be used for this purpose. However, the task of distinguishing a functional key code from a control key code becomes quite complicated if the control keys are placed at these locations. The same advantages can be retained, with easily distinguishable key codes, if a fourth row is added to the matrix. The control keys could then occupy any of the fourteen positions in the fourth row of the 14 x 4 matrix. The key code of a control key would then consist of an encoding of the column location and row location of the key. As in the case of functional keys, a 6-bit code is required to specify the key.

The control key data path is not operational when the stand-alone mode is selected for the calculator terminal. It is only during computer interactions that the data path may be activated, and even then, it is done so at the direction of the software.

The subsystem shown in Figure 4.4 is incorporated into the calculator terminal to regulate the flow of control key information to the computer. When the enable control key output (ECKO) line is activated, the 6-bit key data is allowed to pass to the computer.

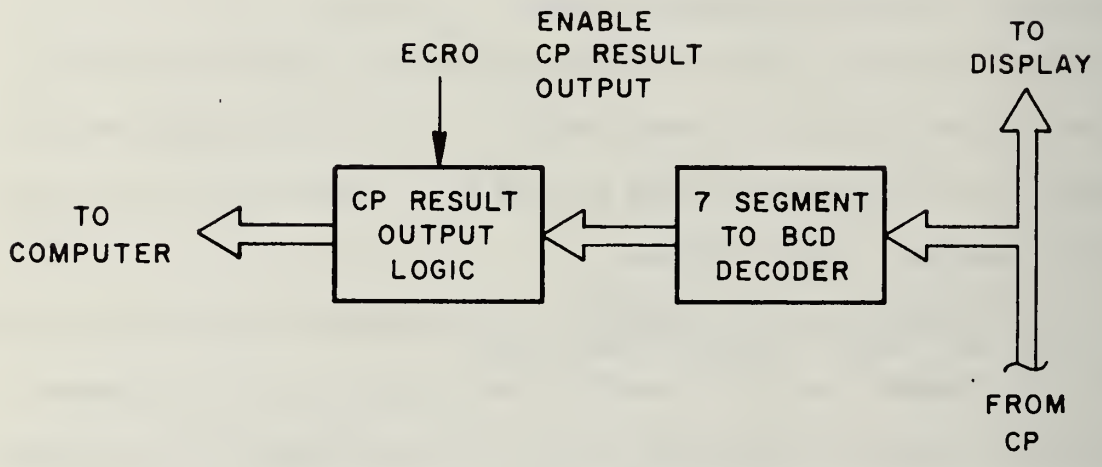
Type 5 - CP to Computer Data

To facilitate shared processing, the type 5 data path is required to allow CP result data to be transmitted to the computer. A simple control module will not suffice in this instance due to the complex nature of the data involved. It will be recalled from Chapter 3 that the CP produces a fourteen digit result in a serial manner and that these results are 7-segment encoded. Hence, a logic network is required to decode the information into the BCD code prior to transmission. Furthermore, the logic network must insure that the digits are transmitted in the proper order. A minimum path width of four bits is required to output the BCD information.



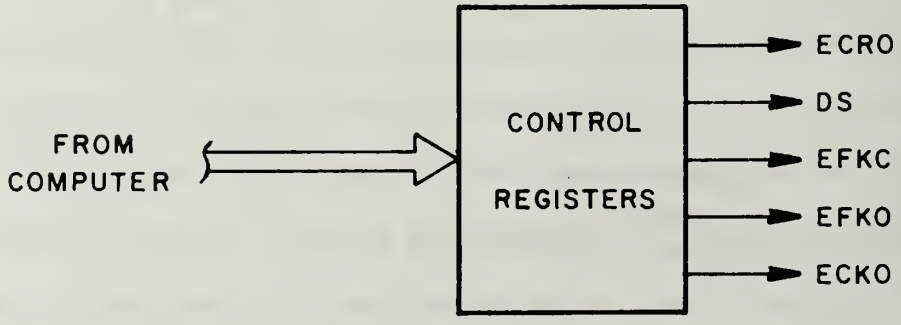
Regulation of Control Key Information into the Computer.

Figure 4.4



Control of CP Display Data Path.

Figure 4.5



Control Registers.

Figure 4.6

Regardless of the nature of the logic network it is necessary for the circuit to know when the data must be transmitted to the computer. An enable CP result output (ECRO) control line can be used for this purpose as shown in Figure 4.5.

Type 6 - Computer to CP Data

To complete the communication link between the computer and the CP it is necessary for the computer to enter data into the CP, a type 6 data transfer. Since this type of information is generated only by the computer, and is not dependent upon a response by the CP or the operator, a control mechanism is not required. Computer to calculator transfers are allowed to proceed without regulation.

Type 7 - Computer to Status Lights

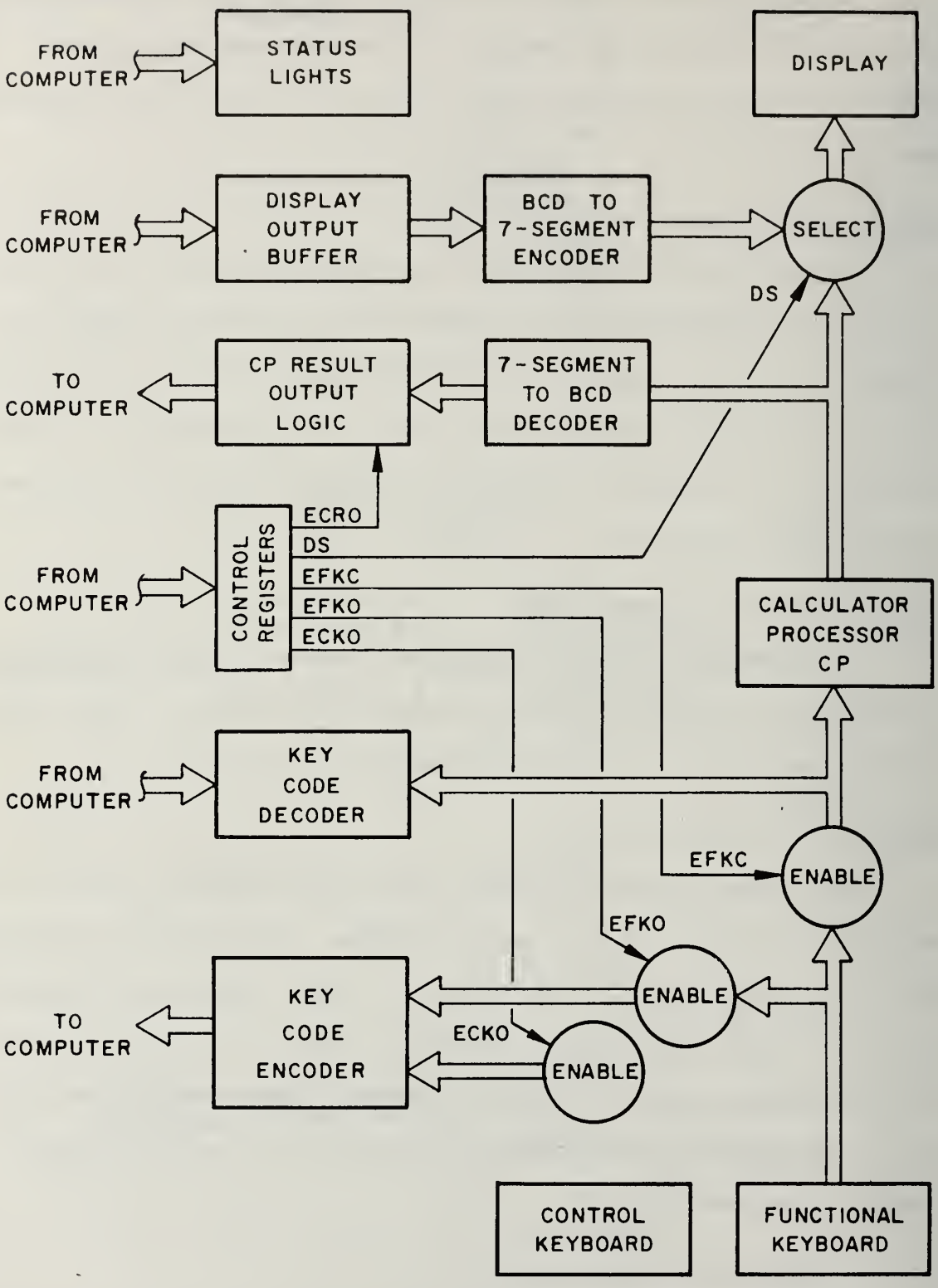
Similarly, data transfers from the computer to the status lights are not dependent upon interactions with the CP or the operator. These types of transmissions are also allowed to proceed without regulation.

Type 8 - Control Register Data

All control signals (DS, ECRO, EFKC, EFKO, and ECKO) have been indicated to be a function of the computer program. The generation of these signals can be accomplished through the use of control registers as indicated in Figure 4.6. This device accepts information from the computer to provide the necessary control signals as outputs. Computer information, transmitted to the control registers, necessitates the addition of a type 8 data path: computer → control registers.

4.3 Communication Synchronization

The control signals, functional modules, and data control modules can be organized as shown in Figure 4.7. The data paths which transfer information between the computer and calculator terminal can be considered as I/O ports at



Block Diagram Showing All Data Paths and Control Modules.

Figure 4.7

this stage in the design process. The remaining design problems involve establishing a communication link between the computer and the CP.

It can be seen from the figure that there are two output channels which transfer information from the calculator to the computer

CP	→	computer	type 5
keys	→	computer	type 3 and 4

Input channels which transfer information from the computer to the calculator include:

computer	→	status lights	type 7
computer	→	display	type 1a
computer	→	control register	type 8
computer	→	CP	type 6

It would not be practical to have an interface circuit for each of the I/O channels listed above. Such a design would greatly increase the complexity of the calculator terminal since all six interfaces would be physically located in the calculator. In addition, this type of design would make the calculator terminal difficult to use over a telephone modem. For these reasons it is felt that communications between the calculator and the computer should be serial, multiplexed transmissions similar to those of a teletype. This design offers simple interface characteristics since only three physical wires need to be connected to the computer: data in, data out, and ground. The system is also easily used over a telephone modem.

Attention is now directed toward designing a communication synchronizing network to allow the six I/O channels access to the data in and data out lines.

Most of the transmitting and receiving requirements can be fulfilled by incorporating a universal asynchronous receiver and transmitter (UART) device into the calculator terminal. This integrated circuit is capable of handling all of the format and serial/parallel conversions which are necessary.

The receiver portion of the UART accepts serial data from the computer and outputs the eight bits of decoded data in a parallel form. An input data ready (IDR) control line is provided to indicate that a data word has been received and is available at the receiver outputs.

The IDR information is used by the subsystem shown in Figure 4.8. The demultiplexing network accepts the eight bits of received data and decodes two of these bits into an input channel address:

- status lights
- display
- control registers
- CP

The remaining six bits are transferred onto the system's input bus. Once the address has been decoded the network generates a load pulse which is sent to a channel and signifies that relevant data may be found on the bus.

- LSL → load status light data
- LDN → load display number data
- LCR → load control register data
- LCI → load CP input data

The process terminates with a clear signal to the UART to reset the receiver and enable further receptions.

The calculator terminal shown in Figure 4.7 indicates that two output channels are required to transfer data to the computer. Since the transmitter portion of the UART is only capable of accepting one eight bit set of data, it is necessary for a multiplexing network to monitor the operations of each channel.

Figure 4.9 shows the multiplexing subsystem which is incorporated into the calculator terminal. The network receives two sets of data and a ready signal from each of the output channels:

-CRR - CP result data ready
-KDR - Key data ready

The network transfers the two sets of data onto one output bus and activates

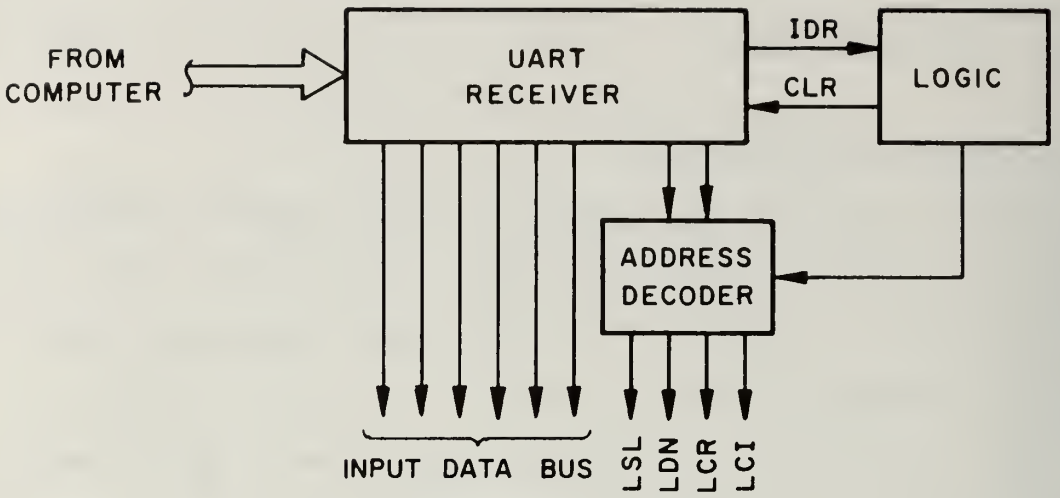
an output data ready (ODR) control line. The transmitter portion of the UART accepts the output data in a parallel manner and begins transmission upon receipt of the ODR command.

4.4 Summary of System Design

Figure 4.10 shows the completed functional diagram of the calculator terminal. The system has all of the capabilities discussed earlier. The computer software is written to accept inputs from, and generate output to, the six I/O channels shown in the figure.

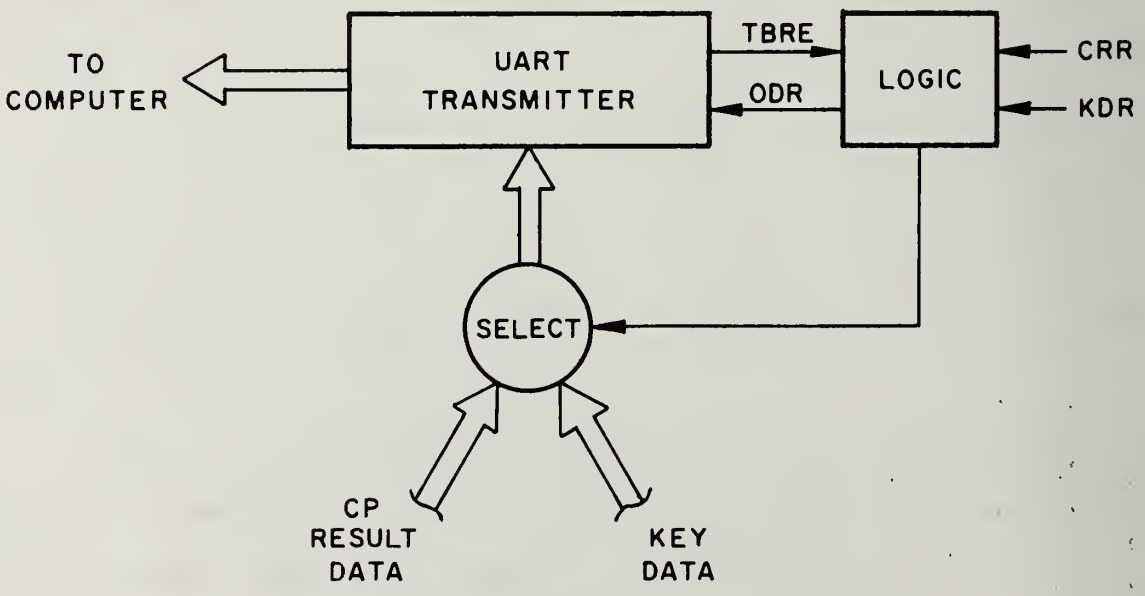
The remainder of the thesis describes the circuit design details as related to each of the subsystems listed below:

1 - CP circuit
2 - display and display selector module
3 - keyboard, key enable modules
4 - computer → CP logic
5 - computer → display
6 - output data multiplexers, keycode logic
7 - status lights, control registers
8 - UART, input data demultiplexers



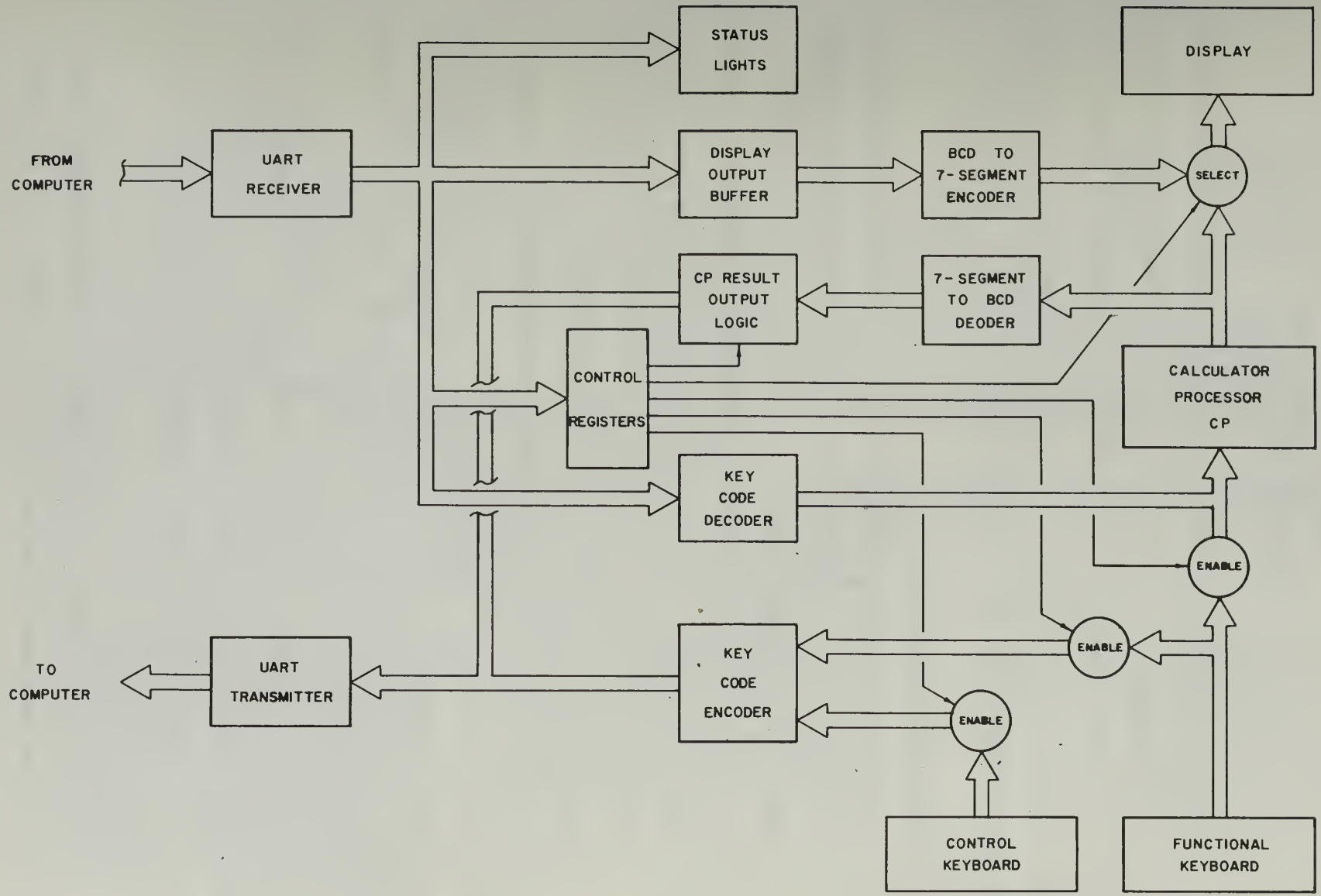
UART Receiver Network.

Figure 4.8



UART Transmitter Network.

Figure 4.9



Complete Block Diagram of the Calculator Terminal.

Figure 4.10

CHAPTER 5

IMPLEMENTATION

Based upon the design concepts presented in Chapter 4 a more detailed description can be given of the circuits used to implement the calculator terminal. It is not the intention of this chapter to describe the purpose of each individual integrated circuit but rather to discuss some of the more difficult design problems and how they were solved.

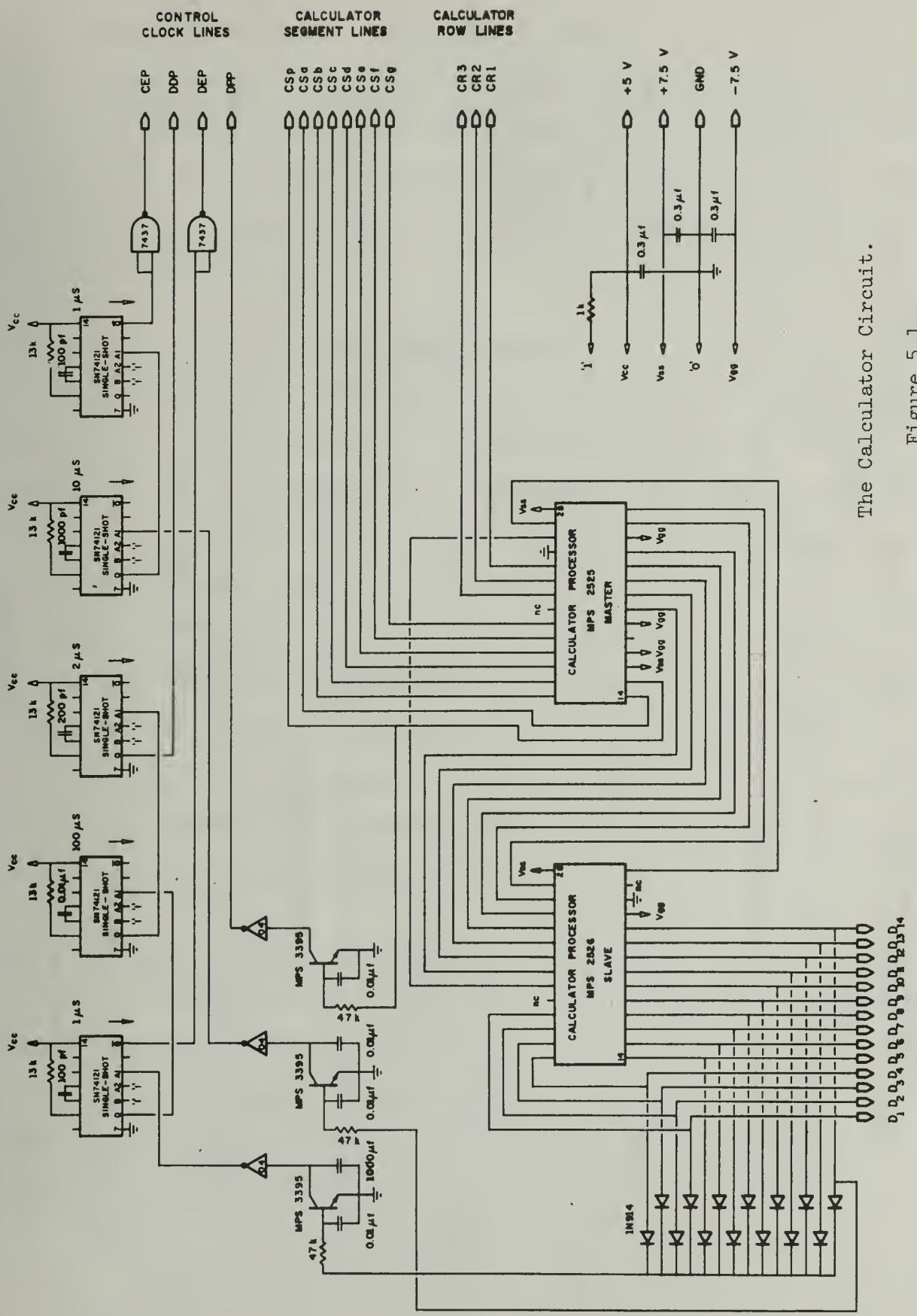
5.1 The Calculator Processor

The circuit of Figure 5.1 illustrates the calculator processing element and the logic circuitry used to generate the control signals. The calculator segment lines (CS_n) provide the only means of accessing CP result data, while the calculator row lines (CR_n) enable key information to enter the CP. Both of these functions are dependent upon the digit lines (D_n).

In order to synchronize the interface with the CP it would be desirable to have access to the clock signals generated by the CP. Since these signals are internal to the CP chips it was found necessary to externally generate the clock signals through the use of the digit lines. An array of fourteen diodes, connected to each digit line, generates the waveform shown in Figure 5.2a. A digit end pulse (DEP), Figure 5.2b, can then be generated as the main synchronizing signal for the entire calculator terminal.

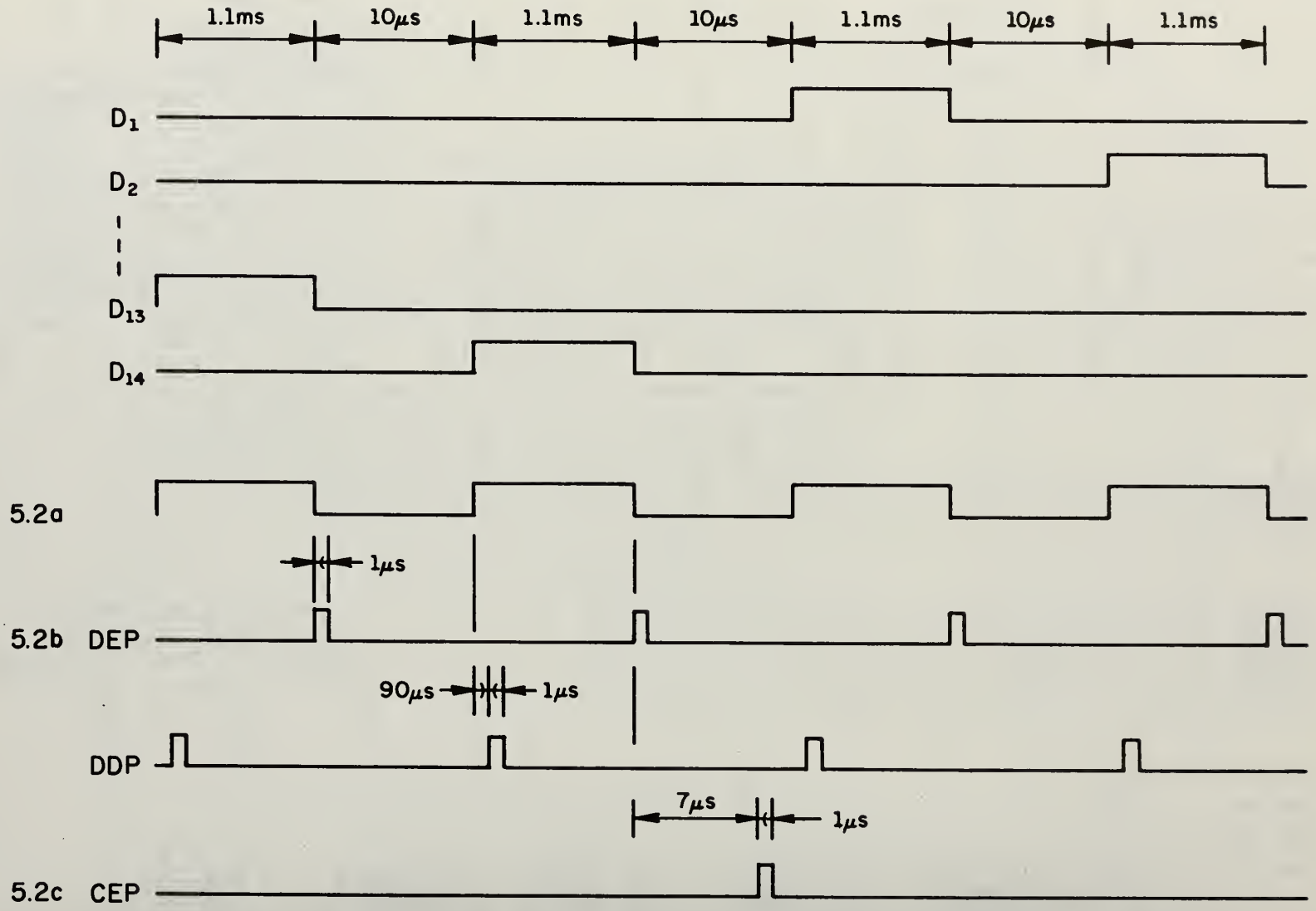
This same waveform also enables the generation of a cycle end pulse (CEP), as shown in Figure 5.2c, which can be used to reset the interface. Similarly, the digit data pulse (DDP) indicates that valid CP result data can be found on the segment lines.

The last clock signal to be considered is the decimal point pulse (DPP). This line provides an indication of the internal state of the CP



The Calculator Circuit.

Figure 5.1



Waveforms on the Digit Lines and Those Generated as Clock Signals.

Figure 5.2

(i.e. executing or display). The absence of a DPP between any two CEP's indicates that the CP is currently executing an instruction. Hence, the next DPP can be interpreted as a "signal done" from the CP. It was learned that pin 10 on the MPS 2525 package can also be used as a "signal done" from the CP. Unfortunately, this feature was discovered too late for inclusion into the system described here.

5.2 Display, Drivers, and Display Selector

A fourteen-digit, seven-segment display provides the only means of output for the computer and the CP. As shown in Figure 5.3, the circuit employs an array of transistors to drive each segment and digit. The digit drivers are controlled by the fourteen CP digit lines while the segment drivers are activated by the selector network.

The selector network accepts two sets of segment information and selects one to send to the display as determined by the state of the display select (DS) control line. As shown, the CP segment data must first be converted to TTL compatible signals to enable its use in the selector. The CP data is also converted from the seven-segment form to the BCD code in order to accomplish transmission to the computer.

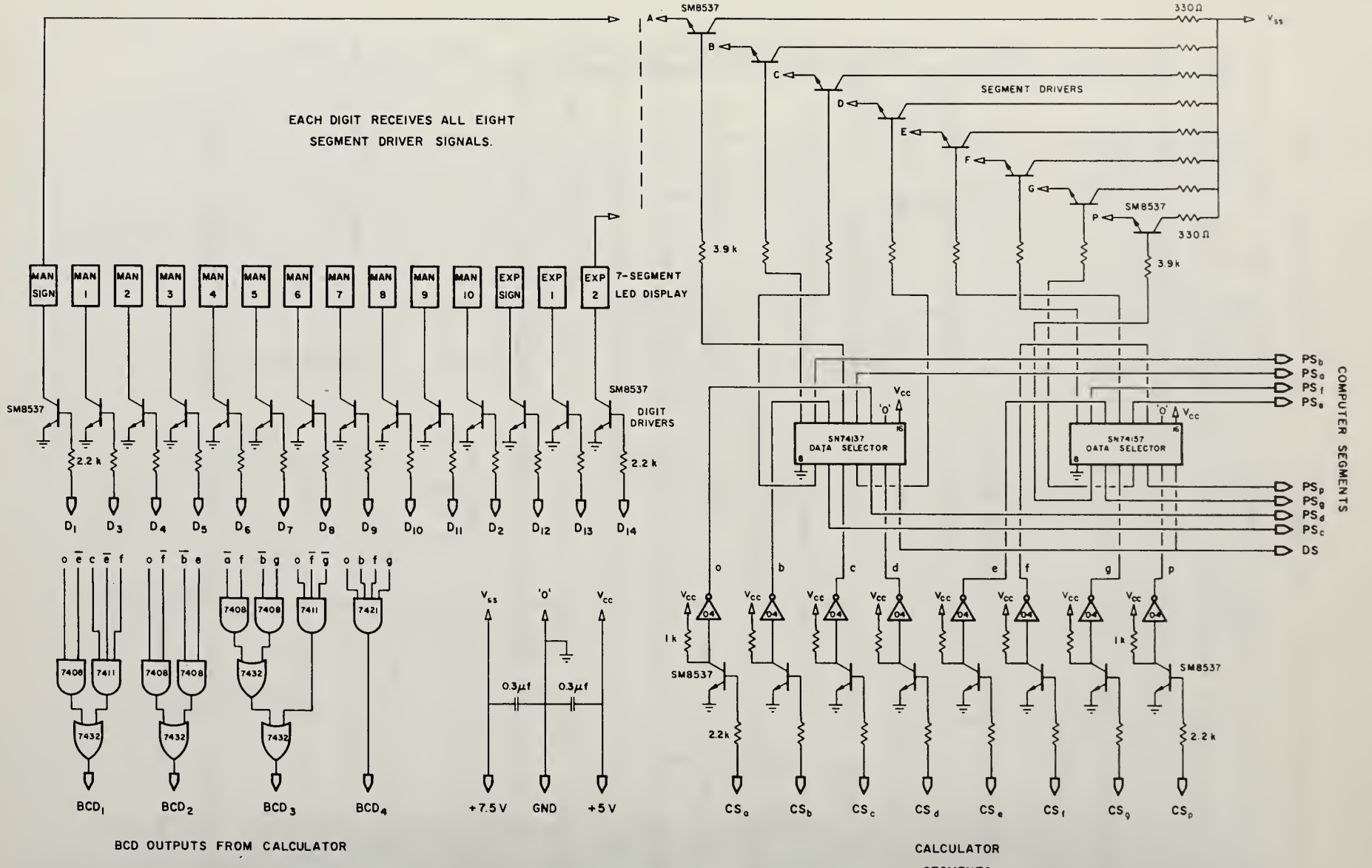
5.3 Key Input Decoder

The network of Figure 5.4 illustrates the circuit used to enable the computer to enter information into the CP:

computer → CP

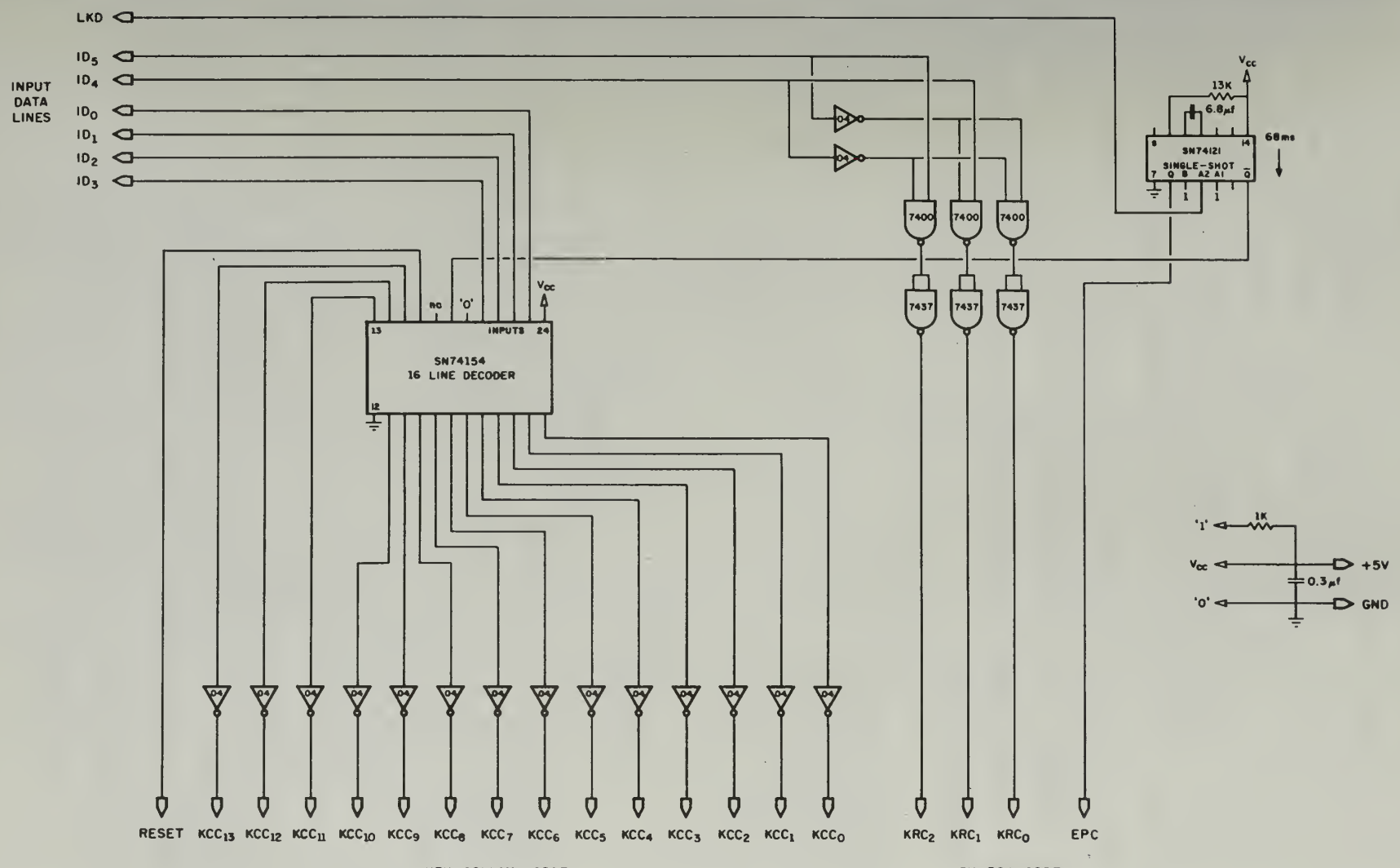
When the computer desires to enter data into the CP it transmits an 8-bit code to the calculator terminal:

OORRCCCC



Display and Display Selector.

Figure 5.3



Key Code Decoder.

Figure 5.4

The first two bits, 00, specify that the data is intended to be used by the key input decoder. This causes a load key data (LKD) pulse which activates the circuit of concern here. The next two bits, RR, are decoded into row signals, KRC_n, for use in the final information entry into the CP. Similarly, the last four bits, CCCC, are decoded into column signals, KCC_n. The column signals, in conjunction with the row signals, uniquely specify a key on the keyboard matrix. The enable processor → CP (EPC) informs the keyboard that the computer is entering data.

The fourteenth column is used to provide a means for the computer to reset the calculator terminal.

5.4 Keyboard and Key Output Control

The network of Figure 5.5 governs the flow of control and functional key information into the CP and the computer. The data paths of concern include:

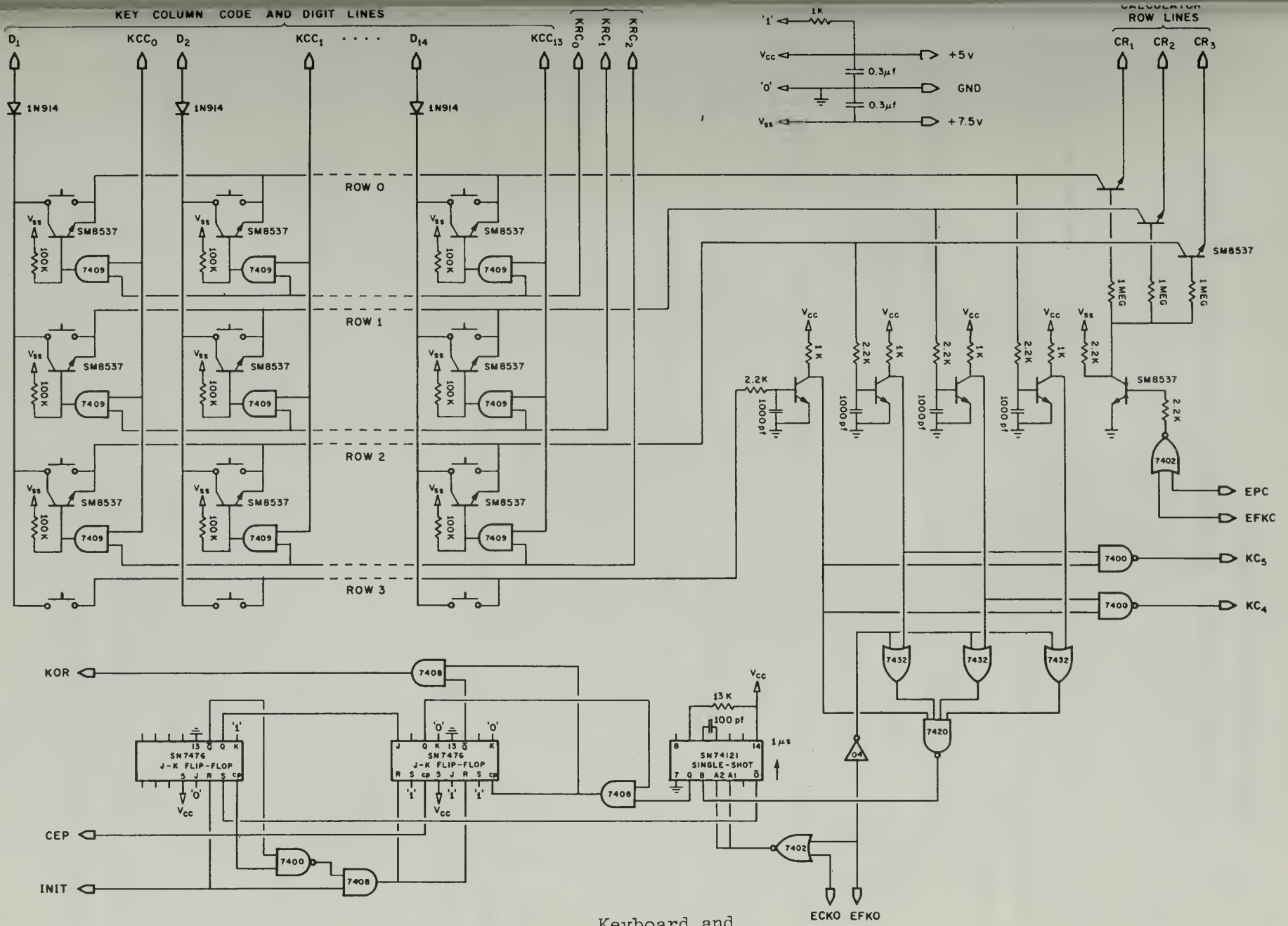
computer	→	CP	EPC
functional key	→	CP	EFKC
functional key	→	computer	EFKO
control key	→	computer	ECKO

The KRC_n and KCC_n lines specify a key which the computer desires to enter into the CP. This action is accomplished by saturating a transistor across the terminals of the key. Note that row 3, which contains only control keys, cannot be activated in this manner.

Functional key information is entered into the CP when either the EPC or EFKC control lines are active.

Key information which is to be sent to the computer is debounced and transmitted upon activation of the key output ready (KOR). The key data is encoded into a row and column specification of the key's location:

$$0, 0, KC_5, KC_4, KC_3, KC_2, KC_1, KC_0$$



Keyboard and
Key Code Encoder.

Figure 5.5

5.5 Display Input from Computer

The computer transfers information to the display in a digit serial manner involving fifteen individual transmissions of 8-bits:

llffdddd

The first two bits, ll, indicate that the data is to be interpreted as display information, this causes activation of the load display information (LDI) control line.

The ff bits are used to specify the action to be taken by the network of Figure 5.6. When ff = 00 the circuit resets and interprets the dddd bits as the decimal point position and dddd is loaded into a 4-bit latch. The next thirteen transmissions will have ff = 01. This causes dddd to be loaded into one word of a 4 x 16 RAM and used as part of the numeric display. The last transmission, fifteenth, has ff = 10. In this case the dddd bits are loaded into the RAM, completing the transfers and the network is allowed to output the data it has sotred.

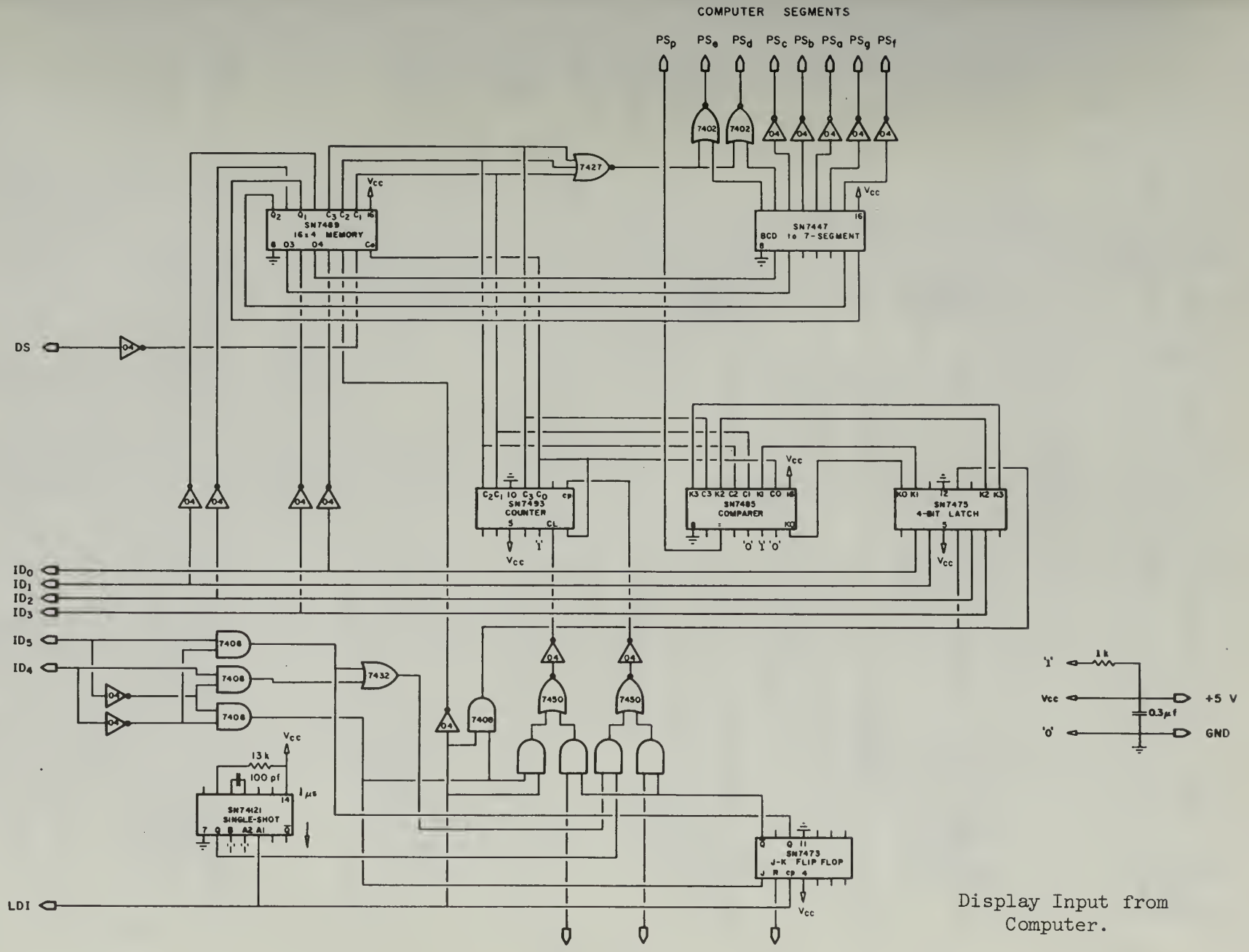
The process of data output involves a constant read cycle through the RAM. A 4-bit BCD code is gated to the 7-segment encoder when the correct digit line appears. This occurrence is determined by an address counter controlled by DEP and CEP.

5.6 CP Result and Key Output

The network of Figure 5.7 governs the flow of all information to the computer. The data types of concern here include:

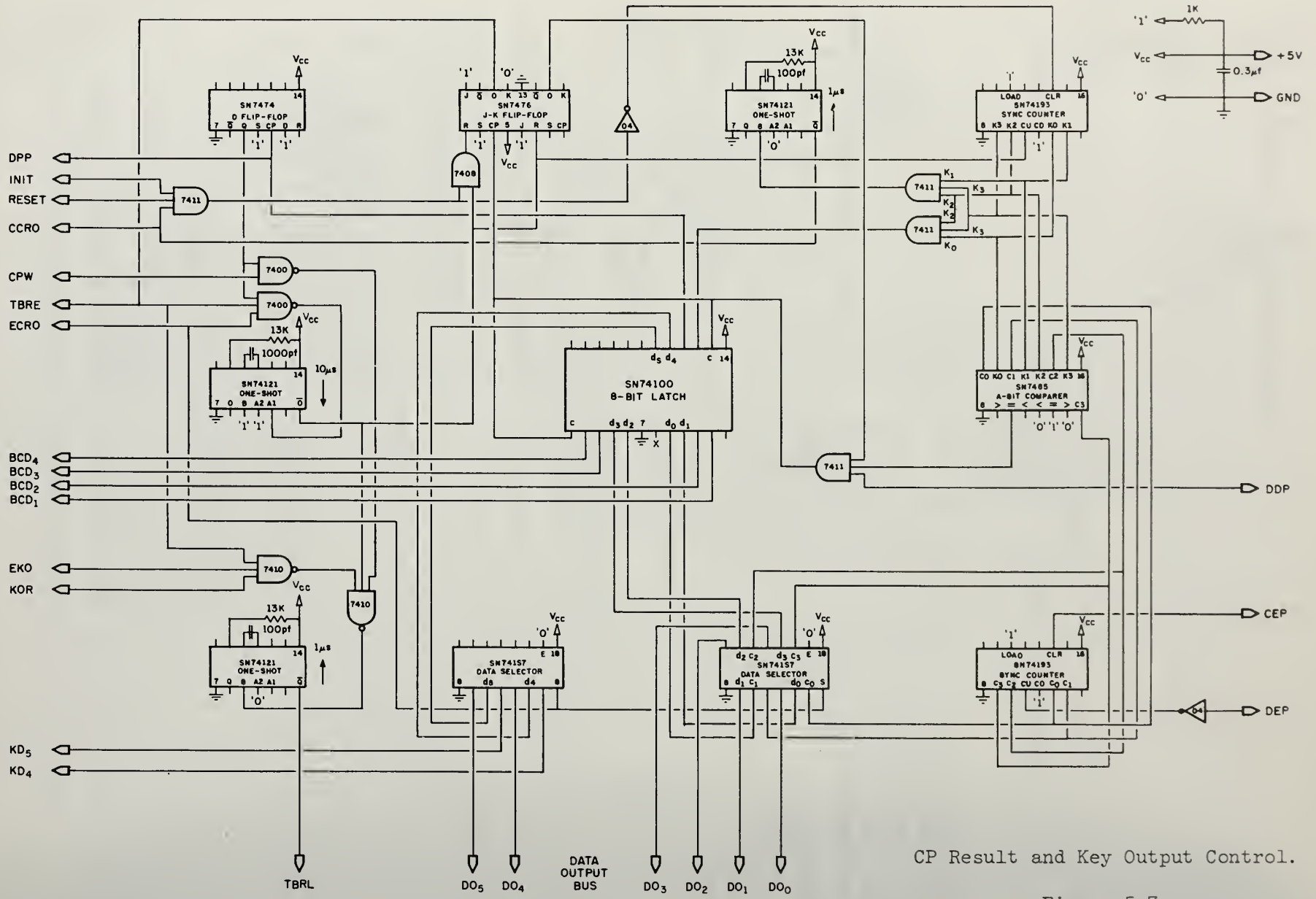
CP → Computer (ECRO)
keys → Computer (EKO)

The key code data is gated onto the data out bus (CO_n) upon reception of a key output ready (KOR) signal from Figure 5.5. KD_5 and KD_4 are accepted externally while KD_3 through KD_0 are generated by an internal counter.



Display Input from Computer.

Figure 5.6



CP Result and Key Output Control.

Figure 5.7

The transmission of the CP's result data occurs upon reception of a CPC signal and involves fourteen separate 8-bit transmissions. The format of the data involved is:

00epdddd

At the nth transmission, the dddd bits represent the BCD encoding of the nth digit. p is high if the decimal point appears in the nth digit. e is activated if n = 14, signaling the end of transmission.

The network also transmits an 8-bit word if the CP wait (CPW) control register is set. The action here is to inform the computer that the CP is ready to accept a command; hence the data bits are unimportant.

All of the above transmissions occur only if the transmitter buffer register is empty (TBRE) and the DPP clock is on. The transfer begins when the circuit activates a transmitter buffer register load (TBRL) signal.

5.7 Control Registers and Status Lights

The network of Figure 5.8 is concerned with the control registers and status lights:

computer → control registers
computer → status lights

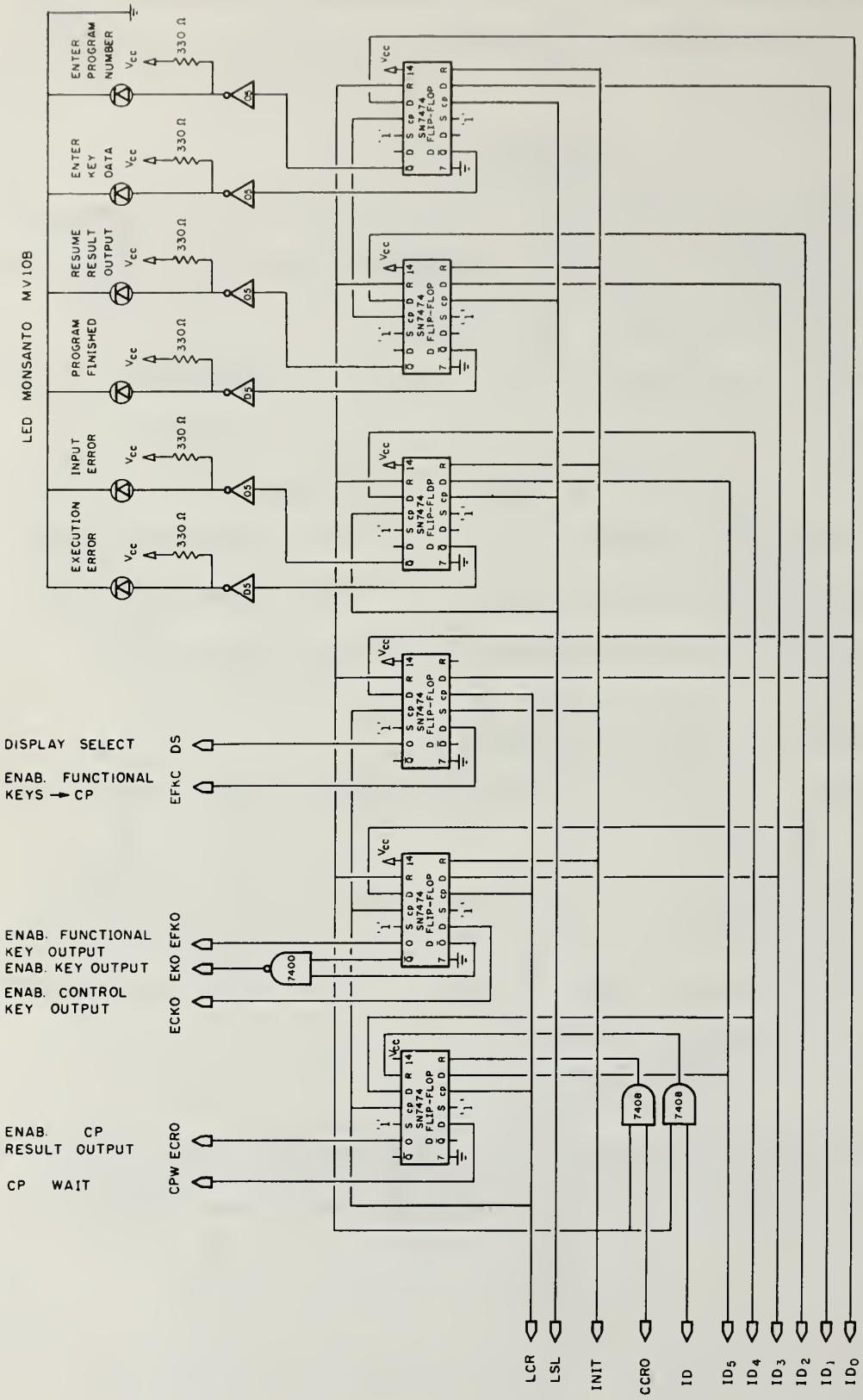
Input data from the computer is loaded into the control registers if the first two bits transmitted are 01:

01dddddd

causing an active value on the load control registers (LCR) control line. Similarly, a 10 on the first two bits activate the load status lights (LSL) control lines:

10dddddd

In both cases the d bits are loaded directly into the associated flip-flop.



Control Registers and Status Lights.

Figure 5.8

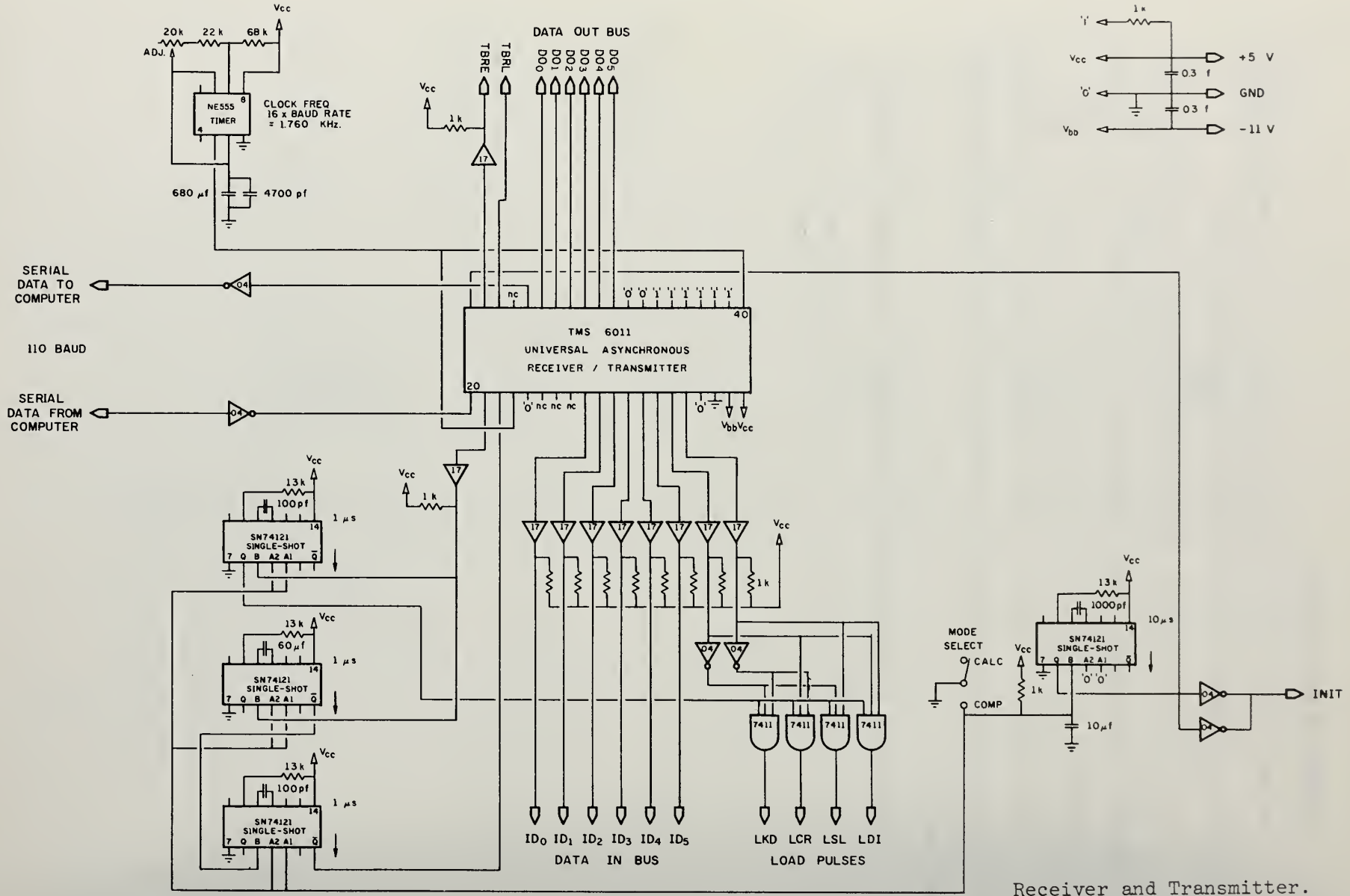
5.8 Transmitter and Receiver

The last circuit to be examined coordinates the I/O between the computer and the calculator terminal as illustrated in Figure 5.9.

The transmitter portion of the UART accepts data from the data out bus when TBRL is activated. The completion of the transmission is signaled by the TBRE line.

Upon reception of an 8-bit word from the computer, the network shown will decode two of these bits into a load pulse LKD, LCR, LSL or LDI as discussed earlier.

The network also provides an initialization pulse (INIT) which is used to reset the entire system upon power-up or a mode change.



Receiver and Transmitter.

Figure 5.9

CHAPTER 6

OPERATION

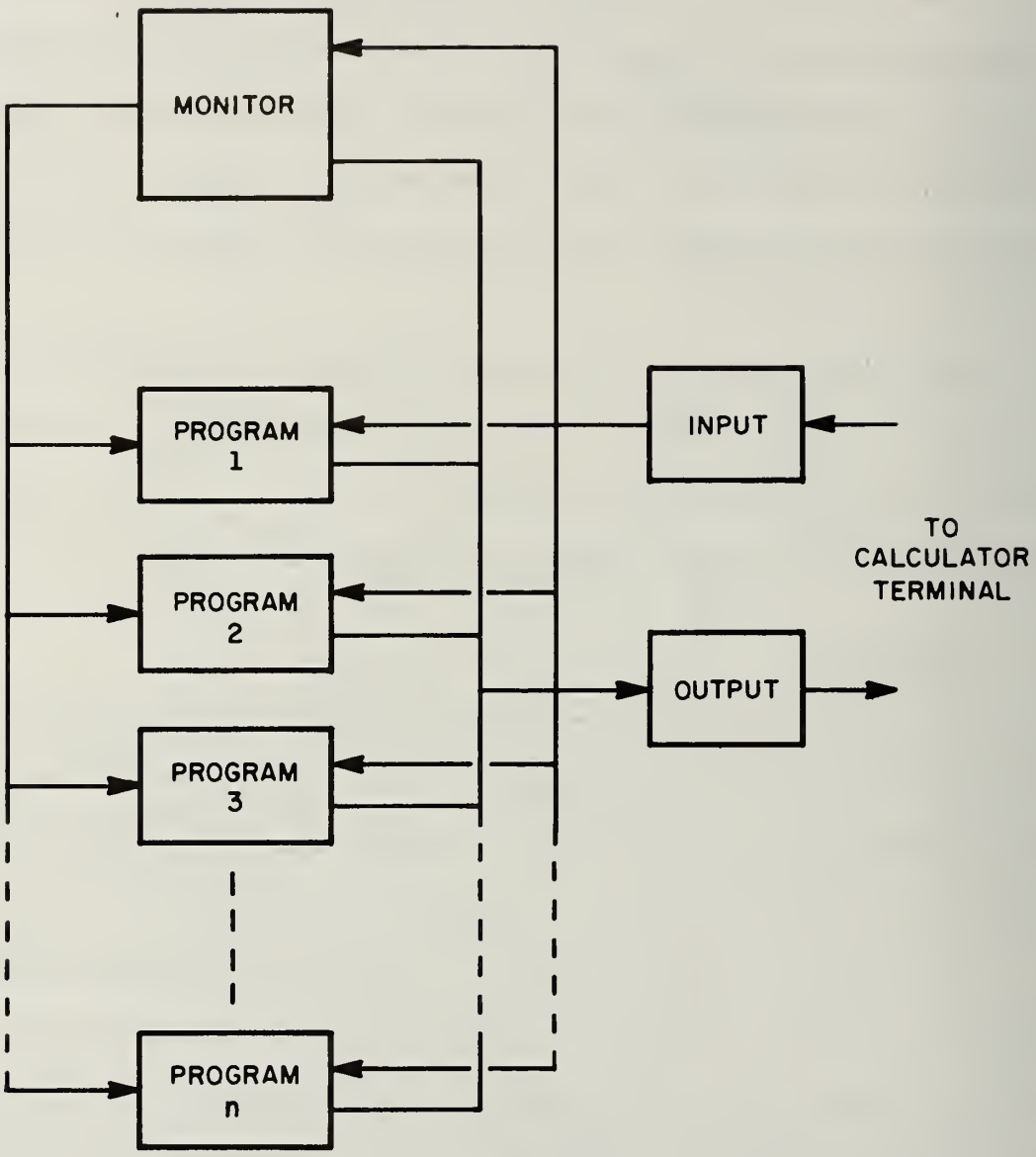
When the calculator terminal is interfaced to a computer it is capable of causing the execution of prewritten programs. Numerical parameters are entered into the computer from the calculator terminal and results are presented on the display. This chapter discusses the operation of the computer/calculator terminal system and outlines some of the difficulties encountered in writing the software.

The capabilities of such a computing system are limited only by the sophistication of the software. Hence, the software concepts presented here are not intended to be rigorous. Indeed, it is through software revisions that the calculator terminal offers its greatest flexibility.

6.1 The Monitor

When the user desires to enter the computer mode of operation he sets the mode switch to COMPUTER and enters the CALL COMPUTER control key. This action activates the service monitor, shown in Figure 6.1, which immediately assumes control of the calculator terminal.

The monitor's initial response is to activate the ENTER PROGRAM NUMBER status light which informs the operator that he should now indicate which program he wishes to call. The program number is entered directly into the CP and is subject to correction by the operator through the use of the CL key. The user's response is terminated by the ENTER PROGRAM NUMBER control key, returning control once again to the monitor. After the monitor has received the program number, and checked its validity, it transfers control to the requested program.



Elementary Flow Chart of Computer Software.

Figure 6.1

It is through the program that the parameter input and result output is accomplished. When the program desires an input parameter it activates the ENTER KEY DATA status light. As before, input information is entered directly into the CP and is correctable in the event an error is made. Control is returned and the data is transferred when the operator enters the ENTER KEY DATA control key.

Once the program has transferred result data to the display it activates the RESUME RESULT OUTPUT status light. This indicates to the operator that valid result data is present on the display. It also indicates that further result outputs are awaiting the entry of the RESUME RESULT OUTPUT control key. Once the last key is entered the program terminates and monitor control resumes.

All responses made by the operator are checked for validity by the monitor and the programs. If it is determined that an error was made, the software responds with an INPUT ERROR status light indicating that the operator must re-enter his response. Furthermore, if an execution error has occurred (i.e. overflow) the program automatically terminates, the monitor is re-entered, and the operator is informed via the EXECUTION ERROR status light.

If, during the input or output phases of a program's execution, the operator desires to terminate the program he may do so through the use of the ABORT control key. The monitor is re-entered and a new program number may be specified. The HALT control key may be entered at any time when the operator no longer wishes to use the computer. This action is completed when the user returns the mode switch to the CALCULATOR position.

6.2 The Program

The software includes a selection of programs written by the operator and called upon to execute a specific function. Two types of programs are apparent for use in a system of this nature. The first of these utilizies the calculating features of the CP and allows shared processing to occur between the CP and the computer. In this mode the CP is acting as a programmable calculator, with the program steps held in the (backup) computer.

Taking advantage of the shared processing features afforded by a system of this nature leads to a program design which consists of sequences of CP key pushes. In this manner a lengthy or tedious mathematical problem can be programmed and executed automatically on the CP.

Numerical parameters are accepted from the CP and encoded into a form which will enable the number to be easily entered into the CP as an operand. This obviously leads to an inefficient use of the computer's memory, but it is felt that the ease of number entry is of greater importance.

It must be remembered during the writing of these programs that the CP is a much slower machine than the computer. In many cases it is possible to saturate the CP with more commands than it can tolerate. It is the responsibility of the person writing the program to recognize when this situation is likely to occur and to force the program into a wait loop until the CP has had a chance to complete its evaluation. The waiting is accomplished by making use of the CP wait feature built into the calculator terminal. The software begins the sequence by transmitting an "are you finished" signal to the CP. If a reply is not received within a reasonable length of time, the software transmits the signal again. This activity proceeds until a reply is received from the CP. The wait cycle is then concluded and the software resumes executing its algorithms.

A second type of program utilizes the I/O capabilities of the calculator terminal and includes only a minimal amount of CP interaction. Programs of this nature might typically make use of the ability of the computer to search through large data files. For these applications it is more efficient to convert the input parameters into a binary format. In this manner table searches and simple arithmetic can be performed directly in the computer.

An example of both types of programs may be found in Appendix A.

Since an alphanumeric display is not available, it is necessary that the operator know the number and the order in which the programs will ask for input parameters. Similarly, the order in which the program will output result data must be known.

CHAPTER 7

SUMMARY AND CONCLUSION

This thesis has suggested a method for interfacing a stand-alone calculator to a computer. The resulting machine, referred to as a calculator terminal, has been shown to function quite well in either mode of operation. The following discussions examine the overall effectiveness of the design and suggest areas where possible improvements may be made.

7.1 Brief Review

The calculator terminal was previously described as a device which is operational as a stand-alone calculator providing the operator with a sufficient set of instructions to facilitate the evaluation of simple arithmetic expressions. At the direction of the operator, the calculator terminal may be interfaced to the computer for the purpose of executing subroutines which are resident in the computer. Control information and data I/O are transmitted to the computer through the calculator terminal; hence, physical control of the machine is not necessary.

Two computing elements, a microprocessor and a commercial calculator chip, were examined for possible use as the calculator processor (CP). It was concluded that a microprocessor design yeilds a system of minimal package count since additional functions could be designed into the system by merely adding additional read-only memory. However, it was also concluded that a calculator element provided the least expensive prototype design due to the fact that the arithmetic capabilities need not be programmed. From these discussions it was determined that a calculator element would best meet the requirements of the CP. MOS Technology's MPS 2525/2526 was chosen for this application.

7.2 Effectiveness of Design

The ability of the calculator terminal to meet the design objectives can be best presented by a separate discussion of the characteristics of the device under the computer and stand-alone modes of operation.

Clearly, the operation of the calculator terminal in the stand-alone configuration is entirely dependent upon the characteristics of the CP. The chosen calculator element provides the operator with a wide selection of trigonometric and logarithmic functions using an algebraic problem entry format. It is felt that this set of operations is amply sufficient for calculating the most common functions.

The calculator terminal has also demonstrated its ability to share processing with a computer. The machine chosen for demonstration purposes was Digital Equipment Company's PDP-11 minicomputer. The calculator terminal was proven operational on both types of programs discussed in Chapter 6. It can then be concluded that at a minimum, the machine will function in a shared processing, remote terminal, or stand-alone environments.

The completed calculator terminal contained a total of 89 IC packages and about 50 transistors. Assuming that there are 50 gates per package, a total gate count of 5000 may be derived. Hence, a pocket version of the device consisting of three or four LSI packages is feasible.

7.3 Design Dependency upon CP and Computer

While the device has been proven operational under the given conditions, it would be helpful to consider the extent to which the design is dependent upon the particular computer and CP selected.

One of the design criteria established at the beginning of the project was to retain teletype compatibility. This implied using a data rate of 110 baud. Since teletype interface hardware exists on most computers, it is

felt that the calculator terminal design is essentially computer independent. Obviously the software will be a function of the computer, but a simple translation is all that should be necessary.

It will be recalled from the initial system design discussions that the calculator terminal system must contain three closed loop information paths:

user ⟷ CP (via functional keys and display)
user ⟷ computer (via control keys, status lights, display)
computer ⟷ CP (via functional key, control registers, CP output)

Obviously, the same information loops must be implemented regardless of the nature of the CP.

An examination into the possible employment of a microprocessor was conducted in Chapter 2. It was found that while this type of CP would yield a minimal package count, the associated costs were too high for initial design work of this nature. However, if this is not a problem, or if the calculator terminal were to be made portable, it would be recommended that this alternative be re-examined. The data paths mentioned earlier would also be implemented in the new design. Only the design details (I/O with the CP) would differ from a calculator terminal design utilizing a commercial calculator element as the CP.

The design details of the system implemented here are obviously a function of the characteristics of the calculator element selected for use as the CP. However, it is not felt that this dependency is critical. As recalled from Chapter 2, most of the calculator elements examined accomplished I/O in a similar manner - via multiplexing, segment, and keyboard row lines. Any design using a different calculator element as the CP must include a method for obtaining results from the multiplexed segment lines and entering information by simulating key pushes. The number of multiplexing and

keyboard row lines is the only variable of question in this case. Hence, the width of the data paths is the only important parameter.

7.4 Performance Enhancement

This thesis introduced the calculator terminal concept which involved the design of a stand-alone calculator which could be interfaced to a computer to facilitate shared processing or remote I/O capabilities. Although the machine was shown to be operational, it would be of benefit to discuss a few modifications which may enhance the overall performance. Three areas can be itemized as topics of this discussion:

- increased speed
- decreased cost (fewer IC's)
- increased capabilities

The characteristics of the calculator terminal are identical to those of the CP when the machine is in the calculator mode of operation. Hence, it is chosen to limit the discussions to performance enhancement when the machine is operating in the computer mode.

7.4.1 Speed Considerations

Perhaps the largest single problem associated with this design is the rather slow speeds of information transfers between the computer and the calculator terminal. The design requirement of using the device in place of a teletype terminal implies the use of a rather slow data transfer rate of 110 baud - one transmission every 100 ms. With this rate of transmission the following time periods can be derived for the various data paths:

1 - computer → display	1500 ms
2 - computer → CP	1600-1846.4 ms
3 - computer → control registers	100 ms
4 - computer → status lights	100 ms
5 - keys → computer	100-115.4 ms
6 - CP → computer	1400-1615.6 ms

Data transfers listed as items 1, 3 and 4 are not dependent upon the CP or CP generated clock lines. Hence, the times associated with these transfers are directly related to the period of the transmitted word. Obviously, the overall system speed cannot be increased by modifying these subsystems.

The reamining items, 2, 5, and 6, are dependent upon the operating clock frequencies of the CP. An increase in speed may be accomplished through correct alterations to these networks. However, this can only be done at the cost of increasing the baud rate.

When entering data into, or obtaining results from, the CP , the interface circuitry must be designed to accommodate the 1.1 ms period of each digit line and the 15.4 ms digit cycle time.

When the computer enters key information into the CP it should activate that key long enough for the CP to recognize and debounce the key entry. Obviously, this time must be greater than one digit cycle, otherwise the CP might not be aware that a key was pushed at all. However, pushing the key for only one digit cycle will only guarantee that one digit pulse is transferred to the keyboard row lines and into the CP. A better design practice would be to push the key for at least two or three digit cycle times, about 30.8 ms, and hope that the CP does not debounce all of the information.

In a similar manner transfers from the keys to computer, and the CP to the computer, are dependent upon the digit cycle time. The maximum speed at which CP result data may be transmitted to the computer is one digit every 15.4 ms for a total transfer time of 215.6 ms.

Assuming that the transmission rate can be increased, the following time limits may be given for the various data paths:

computer	→	display	-
computer	→	CP	246.4-492.8 ms
computer	→	control registers	-
computer	→	status lights	-
keys	→	computer	15.4-30.8 ms
CP	→	computer	215.6-531.2 ms

The above rates assume a transmission time of 15.4 ms per word implying a transfer rate of about 715 baud.

The system may also be speeded up by selecting a calculator processor with a faster cycle time.

7.4.2 Simpler Design

The calculator terminal system described in this thesis was intended to be general enough to enable a large amount of software flexibility. Two of the data paths included in the initial design may be omitted in some applications if a decrease in software flexibility can be tolerated. A 15-20% decrease in the number of integrated circuit packages may be accomplished in this manner, reducing the total package count of the system to about 70.

One of the data paths which may be considered for elimination is:

functional keys → computer

Normally the computer will only be required to accept numeric key entries since the remaining keys (+, 1/x, =, ...) are usually entered directly into the CP. The functional key → CP → computer data paths would then provide the necessary information transfer.

In an analogous manner the computer → display data path may be replaced by a computer → CP → display path. A large number of IC's are eliminated in this case. A further advantage is that the CP could be used to format the numeric output. Hence, the tasks of blanking leading or trailing zeros, and normalizing numbers need not be performed by the software.

In addition to these major revisions a future design may reduce the package count a little further by minimizing gates and similar SSI circuits.

7.4.3 Increased Capabilities

The capabilities of the calculator terminal can be greatly increased by including three additional components into the design.

The present system is greatly limited by the fact that alphanumeric and hard copy outputs are not available. Including such features on future designs would enable the output of character strings. Also, the sometimes annoying task of writing down the CP result data could be eliminated through the use of a printer mechanism.

At the present time the operator is required to write the programs on punched cards or some other medium away from the calculator terminal. The addition of a few keys to the keyboard matrix could resolve this problem by providing the operator with a full alphanumeric keyboard.

REFERENCES

1. Intel Corporation, From CPU to Software, Intel Corporation, Santa Clara, California, 1974.

2. Morris, R. L., and Miller, J. R., Designing with TTL Integrated Circuits, McGraw-Hill Book Company, New York, 1971.

3. MOS Technology, Specification for Scientific Calculator Arrays (MPS 2525-001, MPS 2526-001), MOS Technology, Norristown, Pennsylvania, 1974.

4. Texas Instruments Inc., The TTL Data Book for Design Engineers, Texas Instruments Inc., Dalas, Texas, 1973.

APPENDIX

PROGRAM EXAMPLES

Two program examples are listed in the following pages to illustrate the types of routines which may be executed through the calculator terminal.

Type 1 Program

The first type of program consists of a key pushing routine which is used to evaluate a compound interest equation. The input parameters include:

1 - principal	0 ≤ principal
2 - annual interest rate	0 ≤ interest
3 - number of monthly payments	0 ≤ # payments

which are used to find:

1 - amount of each monthly payment
2 - total of payments
3 - total interest

The equation which is solved is:

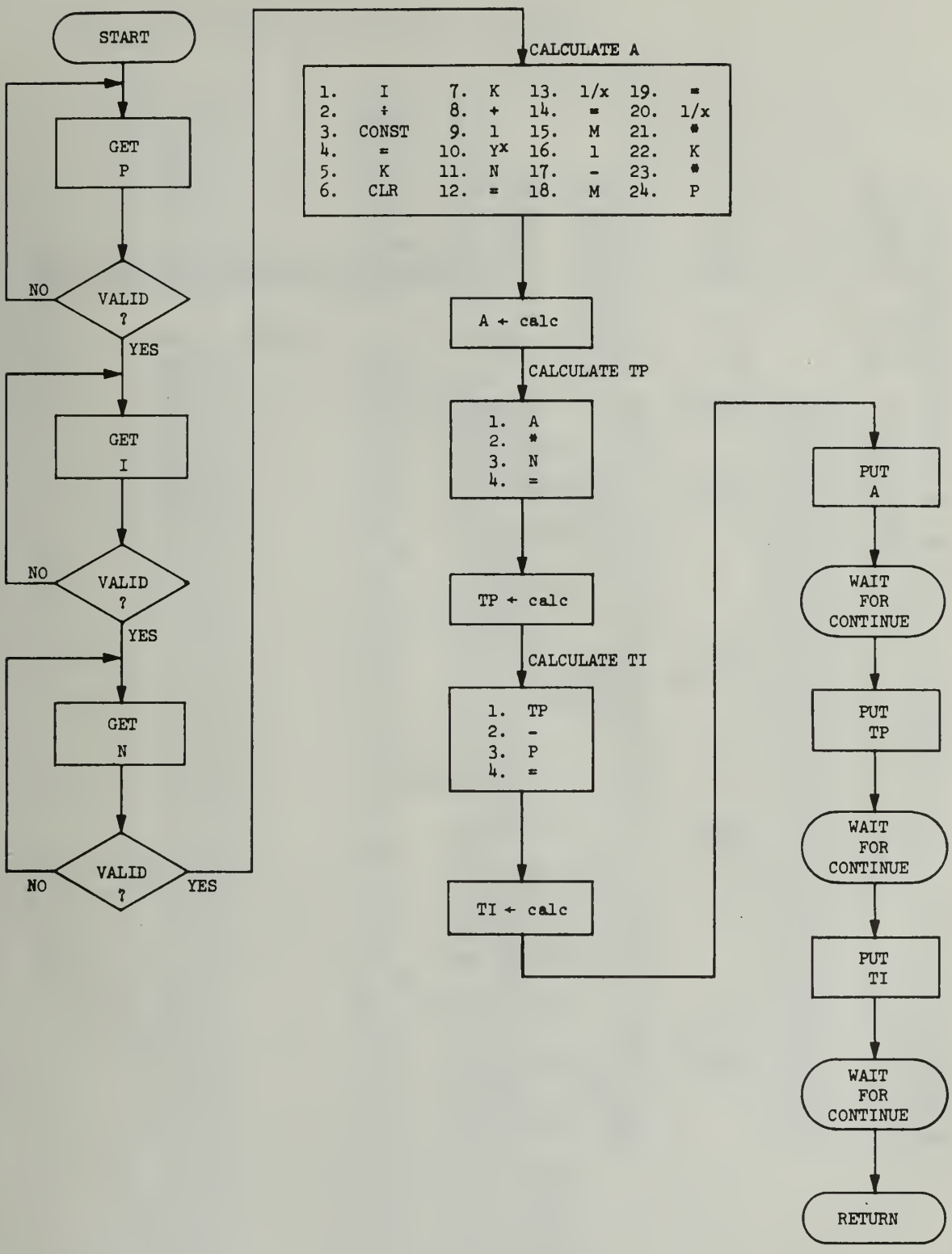
amount of each payment $$A = \frac{k * P}{1 - \left[\frac{1}{1 + k}\right]^{n}}$$

where: P = principal

$$k = \frac{1}{12} * \frac{I}{100} \; ; \qquad I = \text{annual percentage interest rate}$$

n = number of monthly payments

The flow chart of the routine is given in Figure A.1.



Flow Chart of a Key Pushing Routine.

Figure A.1

```
983 7154          P:      .BLKB   30.
984 7212          A:      .BLKB   30.
985 7250          TP:     .BLKB   30.
986 7306          TI:     .BLKB   30.
987 7344     816  CONST:  .BYTE   16,0,0,0,0,0,0,1,2,0,0,20,32,16,0,0,36
988                        .EVEN
989 7366          I:      .BLKB   30.
990 7424          N:      .BLKB   30.
991 7462          K:      .BLKB   30.
992                       ;
993 7520 012767 PROG2:    MOV     #2,PSTAT
994 7526 012767           MOV     #2,ISTAT
995 7534 012767           MOV     #2,NSTAT
996 7542 004567 PIN:      JSR     R5,GET
997 7546 000002 PSTAT:    .WORD   2.
998 7550 007154'          .WORD   P
999 7552 001043           BNE     ERPG2:
1000 554 005767           TST     TBIT
1001 560 003004           BGT     IIN
1002 562 056767           BIS     IERR,PSTAT
1003 570 000764           BR      PIN
1004 572 004567 IIN:      JSR     R5,GET
1005 576 000002 ISTAT:    .WORD   2
1006 600 007366'          .WORD   I
1007 602 001027           BNE     ERPG2:
1008 604 005767           TST     TBIT
1009 610 002004           BGE     NIN
1010 612 056767           BIS     IERR,ISTAT
1011 620 000764           BR      IIN
1012 622 004567 NIN:      JSR     R5,GET
1013 626 000002 NSTAT:    .WORD   2
1014 630 007424'          .WORD   N
1015 632 001013           BNE     ERPG2:
1016 634 005767           TST     TBIT
1017 640 003404           BLE     NON
1018 642 122767           CMPB    #20,N+13
1019 650 001405           BEQ     OK2:
1020 652 056767 NON:      BIS     IERR,NSTAT
1021 660 000760           BR      NIN
1022 662 000207 ERPG2:    RTS     PC
1023 664 004567 OK2:      JSR     R5,TEST
1024 670 007424'          .WORD   N
1025 672 001017           BNE     IPOS
1026 674 004567           JSR     R5,DO
```

```
.MAIN.  MACRO V005C 31-JAN-78 00:10 PAGE 14


1027 700 007154'          .WORD   P
1028 702 000364'          .WORD   DIV
1029 704 007424'          .WORD   N
1030 706 007212'          .WORD   A
1031 710 004567           JSR     R5,CLEAR
1032 714 007306'          .WORD   TI
1033 716 004567           JSR     R5,MOV
1034 722 007154'          .WORD   P
1035 724 007250'          .WORD   TP
1036 726 000167           JMP     PG2DON
1037
1038 732 004567 IPOS:     JSR     R5,DO
1039 736 007366'          .WORD   I
1040 740 000364'          .WORD   DIV
1041 742 007344'          .WORD   CONST
1042 744 007462'          .WORD   K
1043 746 004567           JSR     R5,PUTKEY
1044 752 000340'          .WORD   CL
1045 754 004567           JSR     R5,PUTNUM
1046 760 007462'          .WORD   K
1047 762 004567           JSR     R5,PUTKEY
1048 766 000356'          .WORD   PLUS
1049 770 004767           JSR     PC,CPWAIT
1050 774 004567           JSR     R5,PUTKEY
1051 000 000302'          .WORD   ONE
1052 002 004567           JSR     R5,PUTKEY
1053 006 000370'          .WORD   YTTX
1054 010 004767           JSR     PC,CPWAIT
1055 014 004567           JSR     R5,PUTNUM
1056 020 007424'          .WORD   N
1057 022 004567           JSR     R5,PUTKEY
1058 026 000336'          .WORD   EQ
1059 030 004767           JSR     PC,CPWAIT
1060 034 004567           JSR     R5,PUTKEY
1061 040 000372'          .WORD   INV
1062 042 004767           JSR     PC,CPWAIT
1063 046 004567           JSR     R5,PUTKEY
1064 052 000336'          .WORD   EQ
1065 054 004767           JSR     PC,CPWAIT
1066 060 004567           JSR     R5,PUTKEY
1067 064 000332'          .WORD   M
1068 066 004767           JSR     PC,CPWAIT
1069 072 004567           JSR     R5,PUTKEY
1070 076 000302'          .WORD   ONE
1071 100 004567           JSR     R5,PUTKEY
1072 104 000360'          .WORD   MINUS
1073 106 004767           JSR     PC,CPWAIT
1074 112 004567           JSR     R5,PUTKEY
1075 116 000332'          .WORD   M
1076 120 004767           JSR     PC,CPWAIT
1077 124 004567           JSR     R5,PUTKEY
1078 130 000336'          .WORD   EQ
1079 132 004767           JSR     PC,CPWAIT
1080 136 004567           JSR     R5,PUTKEY
1081 142 000372'          .WORD   INV
1082 144 004767           JSR     PC,CPWAIT
1083 150 004567           JSR     R5,PUTKEY
```

```
.MAIN.  MACRO V005C 31-JAN-78 00:19 PAGE 14


1084 154 000362'          .WORD   MULT
1085 156 004767           JSR     PC,CPWAIT
1086 162 004567           JSR     R5,PUTNUM
1087 166 007462'          .WORD   K
1088 170 004567           JSR     R5,PUTKEY
1089 174 000362'          .WORD   MULT
1090 176 004767           JSR     PC,CPWAIT
1091 202 004567           JSR     R5,PUTNUM
1092 206 007154'          .WORD   P
1093 210 004567           JSR     R5,PUTKEY
1094 214 000336'          .WORD   EQ
1095 216 004767           JSR     PC,CPWAIT
1096 222 004567           JSR     R5,GET
1097 226 000200           .WORD   200.
1098 230 007212'          .WORD   A
1099 232 001047           BNE     EXPG2:
1100 234 004567           JSR     R5,DO
1101 240 007212'          .WORD   A
1102 242 000362'          .WORD   MULT
1103 244 007424'          .WORD   N
1104 246 007250'          .WORD   TP
1105 250 004567           JSR     R5,DO
1106 254 007250'          .WORD   TP
1107 256 000360'          .WORD   MINUS
1108 260 007154'          .WORD   P
1109 262 007306'          .WORD   TI
1110 264 004567 PG2DON:   JSR     R5,PUTDSP
1111 270 007212'          .WORD   A
1112 272 004567           JSR     R5,GET
1113 276 000004           .WORD   4
1114 300 000000           .WORD   0
1115 302 001023           BNE     EXPG2:
1116 304 004567           JSR     R5,PUTDSP
1117 310 007250'          .WORD   TP
1118 312 004567           JSR     R5,GET
1119 316 000004           .WORD   4
1120 320 000000           .WORD   0
1121 322 001013           BNE     EXPG2:
1122 324 004567           JSR     R5,PUTDSP
1123 330 007306'          .WORD   TI
1124 332 004567           JSR     R5,GET
1125 336 000004           .WORD   4
1126 340 000000           .WORD   0
1127 342 001003           BNE     EXPG2:
1128 344 016767           MOV     PGFIN,STATUS
1129 352 000207 EXPG2:    RTS     PC
1130
1131
```

Type 2 Program

A second type of program uses the computer's memory to search through files. The example provided is an inventory system of the 7400 series integrated circuits.

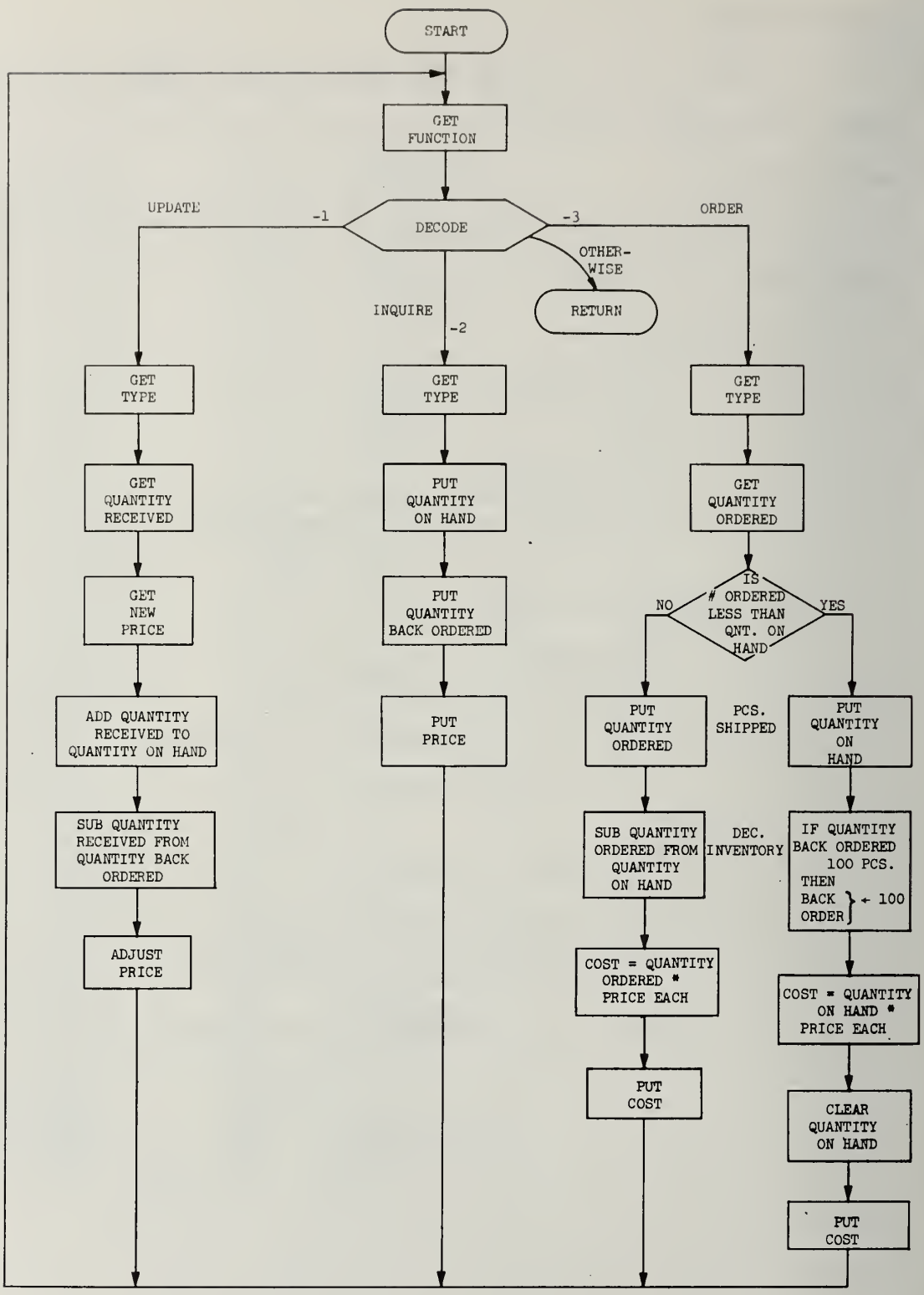
The user may direct the program into one of three directions to perform the desired function:

1 - Place an Order
    A - Given
        - type number
        - number of pieces desired
    B - Action
        - reduce inventory
        - back order parts if necessary
        - output the number of pieces shipped
        - output the cost of this order

2 - Update the Inventory
    A - Given
        - type number
        - number of pieces received
        - new price
    B - Action
        - updates inventory

3 - Inquire
    A - Given
        - type number
    B - Action
        - output number of pieces on hand
        - output number of pieces on back order
        - output current price per piece

The flow chart of the program is given in Figure A.2.



A File Searching Routine.

Figure A.2

```
1132                    ;
1133 354 000000 LIST:   .WORD   000.,100.,0,022.,001.,100.,0,024.
1134 374 000002         .WORD   002.,100.,0,019.,003.,100.,0,024.
1135 414 000004         .WORD   004.,100.,0,024.,005.,100.,0,025.
1136 434 000006         .WORD   006.,100.,0,109.,007.,100.,0,070.
1137 454 000010         .WORD   008.,100.,0,026.,009.,100.,0,027.
1138 474 000012         .WORD   010.,100.,0,026.,012.,100.,0,050.
1139 514 000015         .WORD   013.,100.,0,065.,016.,100.,0,053.
1140 534 000021         .WORD   017.,100.,0,051.,020.,100.,0,021.
1141 554 000027         .WORD   023.,100.,0,051.,025.,100.,0,050.
1142 574 000032         .WORD   026.,100.,0,036.,027.,100.,0,042.
1143 614 000036         .WORD   030.,100.,0,022.,032.,100.,0,032.
1144 634 000045         .WORD   037.,100.,0,045.,038.,100.,0,050.
1145 654 000350         .WORD   040.,100.,0,022.,042.,100.,0,097.
1146 674 000057         .WORD   047.,100.,0,142.,048.,100.,0,106.
1147 714 000062         .WORD   050.,100.,0,024.,051.,100.,0,036.
1148 734 000065         .WORD   053.,100.,0,019.,054.,100.,0,022.
1149 754 000074         .WORD   060.,100.,0,019.,070.,100.,0,053.
1150 774 000110         .WORD   072.,100.,0,030.,073.,100.,0,042.
1151 014 000112         .WORD   074.,100.,0,043.,075.,100.,0,064.
1152 034 000114         .WORD   076.,100.,0,036.,080.,100.,0,058.
1153 054 000121         .WORD   081.,100.,0,197.,082.,100.,0,100.
1154 074 000123         .WORD   083.,100.,0,118.,084.,100.,0,192.
1155 114 000125         .WORD   085.,100.,0,211.,086.,100.,0,040.
1156 134 000131         .WORD   089.,100.,0,474.,090.,100.,0,074.
1157 154 000133         .WORD   091.,100.,0,097.,092.,100.,0,086.
1158 174 000135         .WORD   093.,100.,0,088.,095.,100.,0,094.
1159 214 000141         .WORD   097.,100.,0,324.,100.,100.,0,206.
1160 234 000150         .WORD   104.,100.,0,062.,105.,100.,0,061.
1161 254 000156         .WORD   110.,100.,0,045.,111.,100.,0,061.
1162 274 000171         .WORD   121.,100.,0,045.,122.,100.,0,055.
1163 314 000173         .WORD   123.,100.,0,270.,141.,100.,0,108.
1164 334 000220         .WORD   144.,100.,0,233.,145.,100.,0,146.
1165 354 000226         .WORD   150.,100.,0,333.,151.,100.,0,116.
1166 374 000231         .WORD   153.,100.,0,126.,154.,100.,0,241.
1167 414 000233         .WORD   155.,100.,0,119.,156.,100.,0,116.
1168 434 000235         .WORD   157.,100.,0,104.,160.,100.,0,138.
1169 454 000241         .WORD   161.,100.,0,138.,162.,100.,0,138.
1170 474 000243         .WORD   163.,100.,0,105.,164.,100.,0,151.
1171 514 000245         .WORD   165.,100.,0,156.,166.,100.,0,151.
1172 534 000247         .WORD   167.,100.,0,285.,170.,100.,0,368.
1173 554 000255         .WORD   173.,100.,0,210.,184.,100.,0,175.
1174 574 000271         .WORD   185.,100.,0,266.,190.,100.,0,310.
1175 614 000277         .WORD   191.,100.,0,234.,192.,100.,0,225.
1176 634 000301         .WORD   193.,100.,0,158.,194.,100.,0,125.
1177 654 000303         .WORD   195.,100.,0,100.,196.,100.,0,122.
1178 674 000305         .WORD   197.,100.,0,128.,198.,100.,0,282.
1179 714 000307         .WORD   199.,100.,0,283.,100000,100000,100000,100000
1180                    ;
1181 734        TMP:    .BLKB   30.
1182 772 000000 SBIT:   .WORD   0
1183 774 012136'FUNC:   .WORD   UPDATE,QUIERY,ORDER
1184                    ;
1185 002 005067 PROG3:  CLR     SBIT
1186 006 012767         MOV     #2,FSTAT
1187 014 004567 FGET:   JSR     R5,GET
1188 020 000002 FSTAT:  .WORD   2
1189 022 011734'        .WORD   TMP
1190 024 001040         BNE     ERPG3
1191 026 122767         CMPB    #31,TMP
1192 034 001404         BEQ     OK3
1193 036 056767         BIS     IERR,FSTAT
1194 044 000763         BR      FGET
1195 046 116700 OK3:    MOVB    TMP+12,R0
1196 052 042700         BIC     #177770,R0
1197 056 005300         DEC     R0
```

```
.MAIN.  MACRO V005C 31-JAN-78 00:19 PAGE 14


1198 060 006300          ASL     R0
1199 062 020027          CMP     R0,#6
1200 066 002014          BGE     QUIT
1201 070 004770          JSR     PC,@FUNC(R0)
1202 074 005767          TST     STATUS
1203 100 001012          BNE     ERPG3
1204 102 005767          TST     SBIT
1205 106 001007          BNE     ERPG3
1206 110 012767          MOV     #2,FSTAT
1207 116 000736          BR      FGET
1208 120 016767 QUIT:    MOV     PGFIN,STATUS
1209 126 000207 ERPG3:   RTS     PC
1210
1211                     ;
1212 130 000000 TYPE1:   .WORD   0
1213 132 000000 REC1:    .WORD   0
1214 134 000000 COST1:   .WORD   0
1215                     ;
1216 136 012767 UPDATE:  MOV     #TYPE1,PRM100
1217 144 004567 UPLOOP:  JSR     R5,GET
1218 150 000002          .WORD   2
1219 152 011734          .WORD   TMP
1220 154 001020          BNE     EX1PG3
1221 156 005767          TST     TBIT
1222 162 002413          BLT     ER1PG3
1223 164 004567          JSR     R5,DTOO
1224 170 000000 PRM100:  .WORD   0
1225 172 026727          CMP     PRM100,#COST1
1226 200 001407          BEQ     OK31
1227 202 062767          ADD     #2,PRM100
1228 210 000755          BR      UPLOOP
1229 212 005267 ER1PG3:  INC     SBIT
1230 216 000207 EX1PG3:  RTS     PC
1231 220 004567 OK31:    JSR     R5,LOOK
1232 224 012130          .WORD   TYPE1
1233 226 001401          BEQ     F1
1234 230 000207          RTS     PC
1235 232 005767 F1:      TST     REC1
1236 236 001406          BEQ     SREC
1237 240 066760          ADD     REC1,2(R0)
1238 246 166760          SUB     REC1,4(R0)
1239 254 005767 SREC:    TST     COST1
1240 260 001403          BEQ     SCOST
1241 262 016760          MOV     COST1,6(R0)
1242 270 000207 SCOST:   RTS     PC
1243                     ;
1244 272 000000 TYPE2:   .WORD   0
1245 274         QNT2:   .BLKB   30.
1246 332         ORD2:   .BLKB   30.
1247 370         COST2:  .BLKB   30.
1248                     ;
1249 426 004567 QUIERY:  JSR     R5,GET
1250 432 000002          .WORD   2
1251 434 011734          .WORD   TMP
1252 436 001011          BNE     EX2PG3
1253 440 005767          TST     TBIT
1254 444 002404          BLT     ER2PG3
```

```
.MAIN.  MACRO V005C 31-JAN-78 00:19 PAGE 14


1255 446 004567          JSR     R5,DTOO
1256 452 012272'         .WORD   TYPE2
1257 454 000403          BR      OK32
1258 456 005267 ER2PG3:  INC     SBIT
1259 462 000207 EX2PG3:  RTS     PC
1260 464 004567 OK32:    JSR     R5,LOOK
1261 470 012272'         .WORD   TYPE2
1262 472 001401          BEQ     F2
1263 474 000207          RTS     PC
1264 476 012767 F2:      MOV     #QNT2,PRM102
1265 504 005720 L2:      TST     (R0)+
1266 506 010067          MOV     R0,PRM101
1267 512 004567          JSR     R5,DTOD
1268 516 000000 PRM101:  .WORD   0
1269 520 000000 PRM102:  .WORD   0
1270 522 026727          CMP     PRM102,#COST2
1271 530 001404          BEQ     OKZ
1272 532 062767          ADD     #30.,PRM102
1273 540 000761          BR      L2
1274 542 012767 OKZ:     MOV     #QNT2,PRM103
1275 550 004567          JSR     R5,DPINX
1276 554 012370'         .WORD   COST2
1277 556 004567 L3:      JSR     R5,PUTDSP
1278 562 000000 PRM103:  .WORD   0
1279 564 004567          JSR     R5,GET
1280 570 000004          .WORD   4
1281 572 000000          .WORD   0
1282 574 001332          BNE     EX2PG3
1283 576 026727          CMP     PRM103,#COST2
1284 604 001726          BEQ     EX2PG3
1285 606 062767          ADD     #30.,PRM103
1286 614 000760          BR      L3
1287                     ;
1288 616 000000 TYPE3:   .WORD   0
1289 620 000000 QNT3:    .WORD   0
1290 622 000000 SHIP3:   .WORD   0
1291 624 000000 COST3:   .WORD   0
1292 626        SHIP30:  .BLKB   30.
1293 664        COST30:  .BLKB   30.
1294                     ;
1295 722 004567 ORDER:   JSR     R5,GET
1296 726 000002          .WORD   2
1297 730 011734'         .WORD   TMP
1298 732 001024          BNE     EX3PG3
1299 734 005767          TST     TBIT
1300 740 002417          BLT     ER3PG3
1301 742 004567          JSR     R5,DTOO
1302 746 012616'         .WORD   TYPE3
1303 750 004567          JSR     R5,GET
1304 754 000002          .WORD   2
1305 756 011734'         .WORD   TMP
1306 760 001011          BNE     EX3PG3
1307 762 005767          TST     TBIT
1308 766 002404          BLT     ER3PG3
1309 770 004567          JSR     R5,DTOO
1310 774 012620'         .WORD   QNT3
1311 776 000403          BR      OK33
```

```
.MAIN.  MACRO V005C 31-JAN-70 00:19 PAGE 1+


1312 000 005267 ER3PG3: INC     SBIT
1313 004 000207 EX3PG3: RTS     PC
1314 006 004567 OK33:   JSR     R5,LOOK
1315 012 012616'        .WORD   TYPE3
1316 014 001401         BEQ     F3
1317 016 000207         RTS     PC
1318 020 026067 F3:     CMP     2(R0),QNT3
1319 026 003403         BLE     BO
1320 030 016703         MOV     QNT3,R3
1321 034 000417         BR      ADDIT
1322 036 066760 BO:     ADD     QNT3,4(R0)
1323 044 166060         SUB     2(R0),4(R0)
1324 052 022760         CMP     #100.,4(R0)
1325 060 003403         BLE     NADD
1326 062 062760         ADD     #100.,4(R0)
1327 070 016303 NADD:   MOV     2(R0),R3
1328 074 005067 ADDIT:  CLR     COST3
1329 100 160360         SUB     R3,2(R0)
1330 104 010367         MOV     R3,SHIP3
1331 110 005703 ADDAGN: TST     R3
1332 112 001405         BEQ     DON3
1333 114 066367         ADD     6(R0),COST3
1334 122 005303         DEC     R3
1335 124 000771         BR      ADDAGN
1336 126 004567 DON3:   JSR     R5,OTOD
1337 132 012622'        .WORD   SHIP3
1338 134 012626'        .WORD   SHIP30
1339 136 004567         JSR     R5,OTOD
1340 142 012624'        .WORD   COST3
1341 144 012664'        .WORD   COST30
1342 146 004567         JSR     R5,DPINX
1343 152 012664'        .WORD   COST30
1344 154 004567         JSR     R5,PUTDSP
1345 160 012626'        .WORD   SHIP30
1346 162 004567         JSR     R5,GET
1347 166 000004         .WORD   4
1348 170 000000         .WORD   0
1349 172 001302         BNE     ER3PG3
1350 174 004567         JSR     R5,PUTDSP
1351 200 012664'        .WORD   COST30
1352 202 004567         JSR     R5,GET
1353 206 000004         .WORD   4
1354 210 000000         .WORD   0
1355 212 000207         RTS     PC
1356                    ;
1357 214 012700 LOOK:   MOV     #LIST,R0
1358 220 027510 RLOOK:  CMP     @(R5),(R0)
1359 224 001410         BEQ     FINDIT
1360 226 062700         ADD     #10,R0
1361 232 005710         TST     (R0)
1362 234 002371         BGE     RLOOK
1363 236 005725         TST     (R5)+
1364 240 005267         INC     SBIT
1365 244 000205         RTS     R5
1366 246 005725 FINDIT: TST     (R5)+
1367 250 005067         CLR     TMP
1368 254 000205         RTS     R5
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-75-706	2.	3. Recipient's Accession No.

4. Title and Subtitle

THE DESIGN AND IMPLEMENTATION OF A CALCULATOR TERMINAL

5. Report Date: March, 1975

6.

7. Author(s): John Patrick Sheehan

8. Performing Organization Rept. No. UIUCDCS-R-75-706

9. Performing Organization Name and Address

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Address

13. Type of Report & Period Covered: technical

14.

15. Supplementary Notes

16. Abstracts

This thesis describes an investigation into the possibility of interfacing a commercial calculator chip to a computer. The resulting machine, referred to as a calculator terminal, enables the solution of simple arithmetic problems in a stand-alone mode. More complex problems or those requiring sequences of instructions may be performed by connecting the calculator terminal to the computer. Several processing elements are examined for possible use in this application. The investigation concludes with a description of a possible calculator terminal design and the associated computer software.

17. Key Words and Document Analysis. 17a. Descriptors

Calculator Terminal
Calculator Element
Microprocessor
Data Paths
Control Modules

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 88
	20. Security Class (This Page) UNCLASSIFIED	22. Price

USCOMM-DC 40329-P71